AN ANALOG ARCHITECTURE FOR AUDITORY FEATURE EXTRACTION AND RECOGNITION

A Dissertation
Presented to
The Academic Faculty

By

Paul Devon Smith

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
November 2004

AN ANALOG ARCHITECTURE FOR AUDITORY FEATURE EXTRACTION AND RECOGNITION

Approved by:

Dr. Paul E. Hasler, Advisor
Professor, School of ECE
Georgia Institute of Technology

Dr. Chin-Hui Lee
Professor, School of ECE
Georgia Institute of Technology

Dr. David V. Anderson
Professor, School of ECE
Georgia Institute of Technology

Dr. Bradley A. Minch
Professor, School of ECE
Franklin W. Olin College of Engineering

Dr. Robert J. Butera
Professor, School of ECE
Georgia Institute of Technology

Date Approved: August 2004

DEDICATION

To my family,

for continued support and guidance;

and to the eternal creator

without whom this journey would have never started,

nor been completed.

ACKNOWLEDGEMENTS

I wish to thank my colleagues in the Integrated Computational Electronics lab for their encouragement and support.

PREFACE

Speech recognition systems have been implemented using a wide range of signal processing techniques including neuromorphic/biological inspired and Digital Signal Processing techniques. Neuromorphic/biologically inspired techniques, such as silicon cochlea models, are based on fairly simple yet highly parallel computation and/or computational units. While the area of digital signal processing (DSP) is based on block transforms and statistical or error minimization methods.

Essential to each of these techniques is the first stage of extracting meaningful information from the speech signal, which is known as feature extraction. This can be done using biologically inspired techniques such as silicon cochlea models, or techniques beginning with a model of speech production and then trying to separate the the vocal tract response from an excitation signal. Even within each of these approaches, there are multiple techniques including cepstrum filtering, which sits under the class of Homomorphic signal processing, or techniques using FFT based predictive approaches. The underlying reality is there are multiple techniques that have attacked the problem in speech recognition but the problem is still far from being solved. The techniques that have shown to have the best recognition rates involve Cepstrum Coefficients for the feature extraction and Hidden-Markov Models to perform the pattern recognition.

The presented research develops an analog system based on programmable analog array technology that can perform the initial stages of auditory feature extraction and recognition before passing information to a digital signal processor. The goal being a low power system that can be fully contained on one or more integrated circuit chips. Results show that it is possible to realize advanced filtering techniques such as Cepstrum Filtering and Vector Quantization in analog circuitry. Prior to this work, previous applications of analog signal processing have focused on vision, cochlea models, anti-aliasing filters and other single component uses. Furthermore, classic designs have looked heavily at utilizing op-amps

as a basic core building block for these designs. This research also shows a novel design for a Hidden Markov Model (HMM) decoder utilizing circuits that take advantage of the inherent properties of subthreshold transistors and floating-gate technology to create low-power computational blocks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As portable electronics continue to advance, greater signal processing is required at lower power and at faster rates. While digital systems and DSPs are highly programmable and easy to use, the high power consumption quickly drains batteries and the slow processing can exclude these from real-time situations. However, instead of looking to the digital world for all of life's answers, analog blocks should be used when there are clear wins. Specifically, MOSFETs running in the subthreshold regime can be used to create extremely low-power, continuous-time systems. Plus, with the addition of floating-gate MOSFETs within this analog circuitry, these analog systems take on one of the main characteristics of digital circuitry, which is ease of use through programmability.

This thesis outlines current efforts in creating cooperative analog/digital signal processing (CADSP) systems towards auditory signal processing applications [2, 25]. We discuss current analog circuit approaches towards the front-end signal processing. We discuss our current IC approaches using this technology for an analog signal processing front-end system for speech recognition.

New advances in analog VLSI circuits have made it possible to perform operations that more closely reflect those done in DSP applications, or that are desired in future DSP applications. Further, analog circuits and systems can be *programmable*, reconfigurable, adaptive, and at a density comparable to digital memories (for example, 100,000+ multipliers on a single chip) Therefore, one might wonder if we have both digital and analog signal processing (DSP and ASP respectively) available, how does one choose a particular solution for a given auditory application. The question is where to partition the analog–digital boundary, as shown in Figure 1, to enhance the overall functionality of a system by utilizing analog/digital computations in mutually beneficial ways. By adding functionality to our analog systems, we enhance the capabilities of the controlling digital system, and

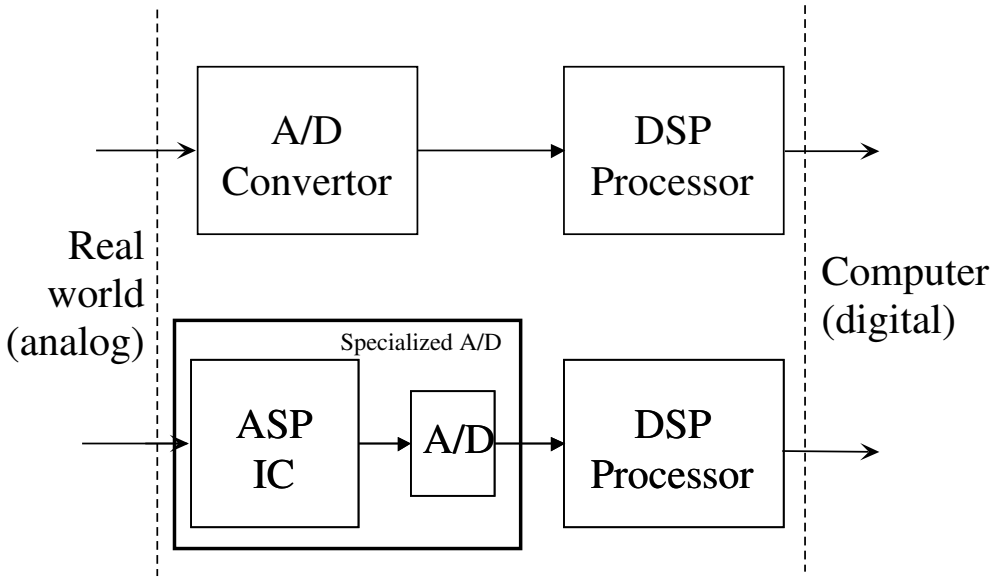


Figure 1. CADSP concept. Computations that have typically been implemented completely in a digital sytem, are being moved to analog implementations for improvements in power and area. In a purely digital system, the bottleneck is in the analog-to-digital conversion. There are also significant gains by reducing the amount of data being transmitted over the analog -to-digital pipe. Accuracy of the data being used in the digital system reduces down to 10 bits of accuracy due to quantization noise and fixed-point errors.

therefore, the entire product under consideration. Further, this additional computational power allows for expansion of current DSP algorithms to incorporate more biologically inspired techniques in its algorithms.

Speech recognition systems have been implemented using a wide range of signal processing techniques including neuromorphic/biological inspired [44, 45, 47, 85] and Digital Signal Processing techniques [65, 9, 74, 16, 86, 87]. Neuromorphic/biologically inspired techniques such as silicon cochlea models are based on simple and highly parallel computation and computational units. While in the area of digital signal processing (DSP) is based on block transforms and statistical or error minimization methods. In addition to this, the computation occurs serially. Over the past years DSP has grown into a very popular topic. Applications using DSPs have also grown fuelled by the increase in computing power and decrease in cost, which has been a very positive step, particularly in the area of speech recognition.

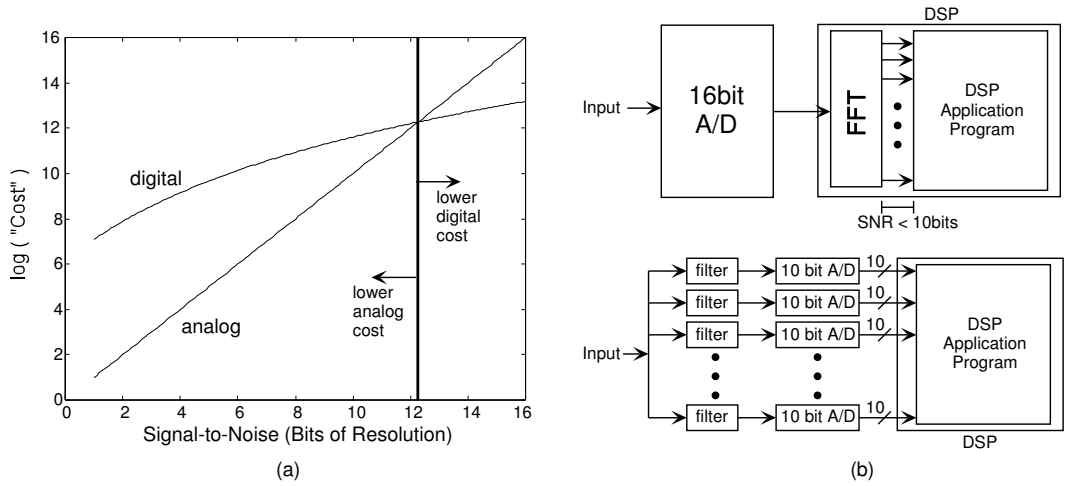


Figure 2. Partitioning a system between analog and digital implementations is critical. There is a point where one domain is more efficient based on costs factors such as die-area and hours of development.

Essential to each of these techniques is the first stage of extracting meaningful information from the speech signal, which is known as feature extraction. This can be done using biologically inspired techniques such as silicon cochlea models, or techniques beginning with a model of speech production as a convolution of the vocal tract with an excitation signal and then trying to separate the signals. Even within each of these approaches, there are multiple techniques including cepstrum filtering, which sits under the class of Homomorphic signal processing, or techniques using classic FFT based predictive approaches. The underlying reality is there are multiple techniques that have attacked the problem is speech recognition and the problem is still far from being solved. The techniques that have shown to have the best recognition rates involve Cepstrum Coefficients for the feature extraction and Hidden-Markov Models to perform the pattern recognition.

The proposed research will develop an analog system based on floating-gate technology that can perform the initial stages of auditory feature extraction and recognition before passing information to a digital signal processor. This concept is illustrated in Figure 4. The research aims to develop a low power system that can be fully contained on one or more integrated circuit chips. Preliminary results show that it is possible to realize an advanced filtering technique known as Cepstrum Filtering and a Vector Quantization technique, in

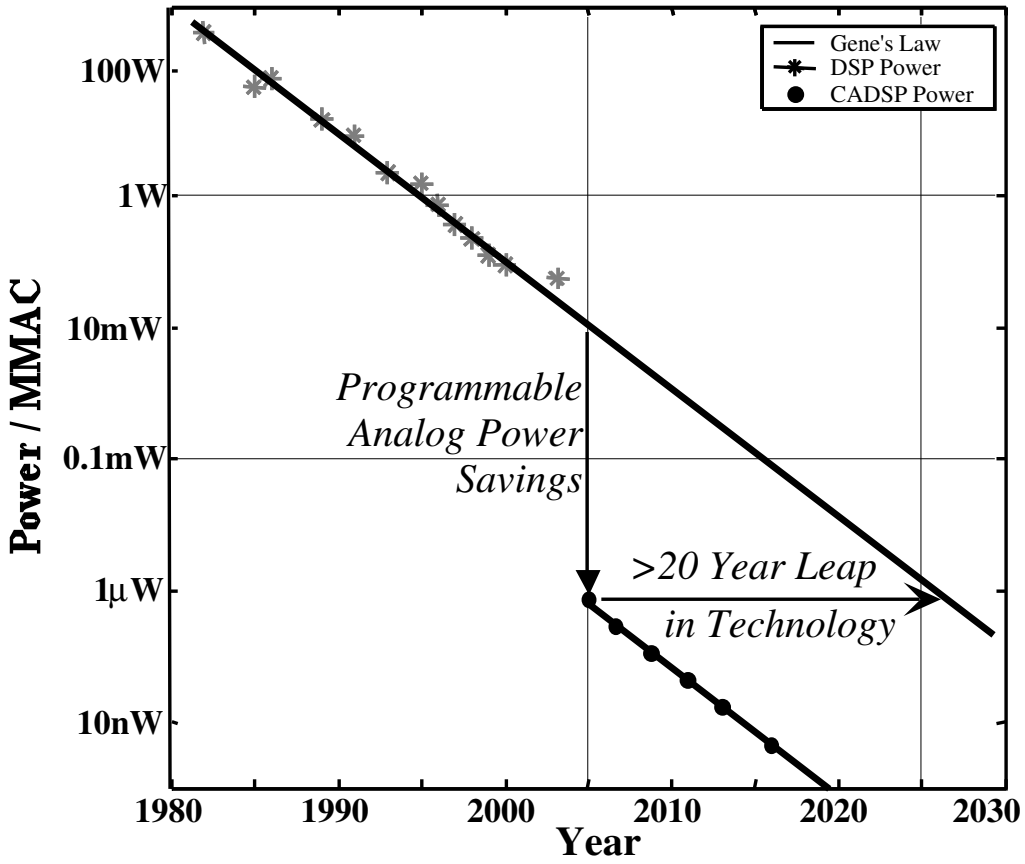


Figure 3. DSP power/MMAC vs. Time: DSP power per MMACs has tracked Moore's Law for over two decades [14]. Current limitations for high-end systems are imposed by the Analog-To-Digital Converter. This bottleneck can be removed by shifting some of the processing from the digital domain to the analog domain. Results show an equivalent of up to a 20 year increase in power per MMAC of computation by implementing Cooperative-Analog Digital Signal Processing Techniques.

analog circuitry. Prior to this work, previous applications of analog signal processing have focused on vision, cochlea models, anti-aliasing filters and other single component uses. Furthermore, classic designs have looked heavily at utilizing op-amps as a basic core building block for these designs. This research also proposes a novel design for a Hidden Markov Model (HMM) decoder. This project will utilize circuits that take advantage of the inherent properties of subthreshold transistors and floating-gate technology to create low-power computational blocks.

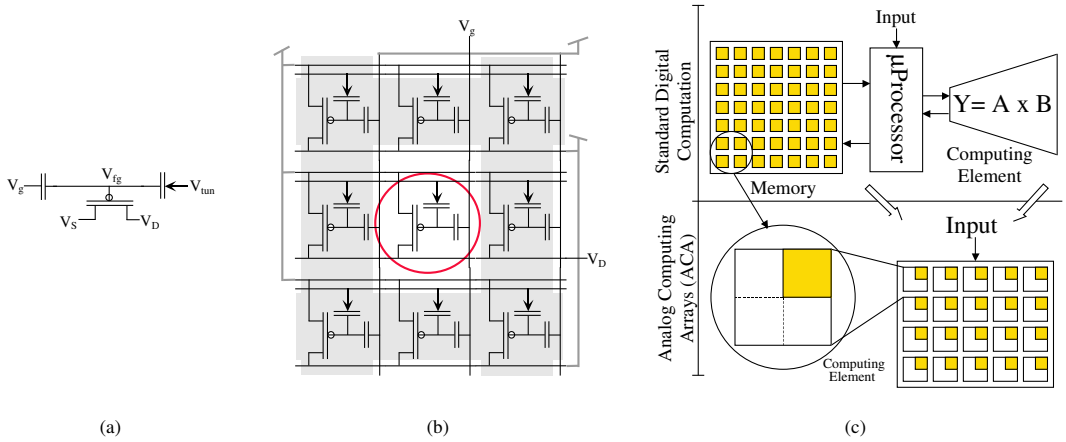


Figure 4. Development of Analog computational arrays. a.) Micro-level: single floating-gate transistor element. b.) Macro-level: Array of floating-gate elements. c.) System-level: Analog Computing Arrays are a combination of computing and memory elements as detailed in the digital system. This consolidation reduces overall die area and substantially reduces overall system power.

1.1 Physiology of the human ear

The human ear is an exceedingly complex organ. Figure 5 illustrates the major structures and processes that comprise the human ear. The outer ear is composed of two parts, the visible flap of skin and cartilage attached to the side of the head, and the ear canal, a tube about 0.5 cm in diameter extending about 3 cm into the head. These structures direct environmental sounds to the sensitive middle and inner ear organs located safely inside of the skull bones. Stretched across the end of the ear canal is a thin sheet of tissue called the tympanic membrane or ear drum. Sound waves striking the tympanic membrane cause it to vibrate. The middle ear is a set of small bones that transfer this vibration to the cochlea (inner ear) where it is converted to neural impulses. The cochlea is a liquid filled tube roughly 2 mm in diameter and 3 cm in length. The cochlea is curled up and looks like a small snail shell as shown in Figure 5. In fact, cochlea is derived from the Greek word for snail.

When a sound wave tries to pass from air into liquid, only a small fraction of the sound is transmitted through the interface, while the remainder of the energy is reflected. This is because air has a low mechanical impedance (low acoustic pressure and high particle

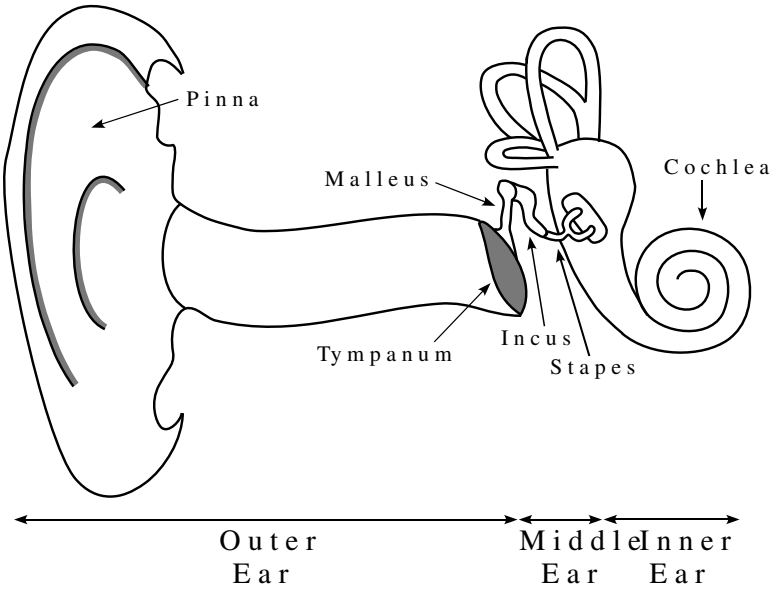


Figure 5. Physiology of the outer ear.

velocity resulting from low density and high compressibility), while liquid has a high mechanical impedance. In less technical terms, it requires more effort to wave your hand in water than it does to wave it in air. This difference in mechanical impedance results in most of the sound being reflected at an air/liquid interface.

The middle ear is an impedance matching network that increases the fraction of sound energy entering the liquid of the inner ear. For example, fish do not have an ear drum or middle ear, because they have no need to hear in air. Most of the impedance conversion results from the difference in area between the ear drum (receiving sound from the air) and the oval window. The ear drum has an area of about $60mm^2$, while the oval window has an area of roughly $4mm^2$. Since pressure is equal to force divided by area, this difference in area increases the sound wave pressure by about 15 times.

Contained within the cochlea is the basilar membrane, the supporting structure for about 12,000 sensory cells forming the cochlear nerve. The basilar membrane is stiffest near the oval window, and becomes more flexible toward the opposite end, allowing it to act as a frequency spectrum analyzer. When exposed to a high frequency signal, the basilar

membrane resonates where it is stiff, resulting in the excitation of nerve cells close to the oval window. Likewise, low frequency sounds excite nerve cells at the far end of the basilar membrane. This makes specific fibers in the cochlear nerve respond to specific frequencies. This organization is called the place principle, and is preserved throughout the auditory pathway into the brain.

Another information encoding scheme is also used in human hearing, called the volley principle. Nerve cells transmit information by generating brief electrical pulses called action potentials. A nerve cell on the basilar membrane can encode audio information by producing an action potential in response to each cycle of the vibration. For example, a 200 hertz sound wave can be represented by a neuron producing 200 action potentials per second. However, this only works at frequencies below about 500 hertz, the maximum rate that neurons can produce action potentials. The human ear overcomes this problem by allowing several nerve cells to take turns performing this single task. For example, a 3000 hertz tone might be represented by ten nerve cells alternately firing at 300 times per second. This extends the range of the volley principle to about 4 kHz, above which the place principle is exclusively used.

Our impression of the loudness of a sound corresponds better to the log of the acoustic power rather than its linear value. Thus, it is common to express sound intensity on a logarithmic scale, called decibel SPL (Sound Power Level). On this scale, 0 dB SPL is a sound wave power of 10 -16 watts/cm 2 , about the weakest sound detectable by the human ear. Normal speech is at about 60 dB SPL, while painful damage to the ear occurs at about 140 dB SPL. Successively doubling the power in a sound gives an impression of roughly equal steps in loudness. The loudness of a sound is therefore normally expressed on the logarithmic decibel (dB) scale, where a factor of ten increase corresponds to 10 dB. (The cube root of the power is now generally considered to match perceptual loudness even more closely, but the use of the log scale has been retained.)

The difference between the loudest and faintest sounds that humans can hear is about

120 dB, a range of one-million in amplitude. Listeners can detect a change in loudness when the signal is altered by about 1 dB (a 12% change in amplitude). In other words, there are only about 120 levels of loudness that can be perceived from the faintest whisper to the loudest thunder. The sensitivity of the ear is amazing; when listening to very weak sounds, the ear drum vibrates less than the diameter of a single molecule!

The perception of loudness relates roughly to the sound power with an exponent of 1/3. For example, if you increase the sound power by a factor of ten, listeners will report that the loudness has increased by a factor of about two. This scale can work to your disadvantage in the opposite direction when eliminating undesired environmental sounds. An example would be to cover 99% of your wall with a perfect soundproof material in an effort to block unwanted noise. Even though the sound power has been reduced by 99% to only 1% of its former value, the perceived loudness has only dropped by 20%.

The range of human hearing is considered to be 20 Hz to 20 kHz, with a peak in sensitivity to sounds between 1 kHz and 4 kHz. It falls off markedly somewhere below 100Hz and, depending on our age, somewhere above 5 to 10kHz. For example, listeners can detect sounds as low as 0 dB SPL at 3 kHz, but require 40 dB SPL at 100 hertz (an amplitude increase of 100). Listeners can tell that two tones are different if their frequencies differ by more than about 0.3% at 3 kHz. This increases to 3% at 100 hertz. For comparison, adjacent keys on a piano differ by about 6% in frequency.

1.2 The Speech Signal

The organs primarily involved in producing speech are the larynx, visible as the ”Adam’s apple” in men, and which contains a pair of muscular folds called the vocal cords, and the vocal tract, which is a tube leading from the larynx along the pharynx and then branching into the oral cavity leading to the lips and through the nasal cavity to the nostrils.

Acoustic energy in speech can be generated in two different ways. The primary mechanism, known as *voiced* excitation, occurs in the larynx. The muscles in the larynx place

the vocal cords close together and make them loose enough that when air from the lungs is driven through them they open and close quasiperiodically at an average rate of about 110 times a second for a man and about twice that for a woman. The main instant of voiced excitation occurs not, as one might expect, on opening, but when the air flow from the lungs is suddenly stopped as the cords are pulled together by Bernoulli forces resulting from air flow through the opening. The forces are caused by a pressure drop and change of flow direction. Bernoulli Forces tend to close and constrict any form of valve opening and limit the controllable power through the valve [32]. The resulting voiced speech sounds include all vowels (unless whispered) and many consonant sounds: the words *Roman, yellow*, and *wiring*, for example, are composed entirely of voiced sounds.

In the second mechanism for generating acousitc energy in speech air passes from the lungs through the larynx with the vocal cords held apart and is forced through a constriction formed by the tongue or lips causing turbulence and resulting in a aperiodic, noise-like excitation. Sounds generated purely in this way (such as ”s” and ”ft” in soft) are said to be voiceless, and they generally play a less important role in speech than voiced sounds.

Vowel sounds are voiced. They are produced without any obstruction in the oral cavity. If the branch to the nasal tract is open, the vowel is said to be nasalized (such as the vowels in the French words *bon, sans, faim*, etc.). Vowels can be further divided into so-called pure vowels, which can be produced in isolation with a stationary vocal tract, and *diphthongs*, (such as in the words *say, sow* and *sigh*) where a movement of the articulators (the tongue, lips or jaw) is necessary.

In contrast to vowels, consonants always involve a narrowing in the oral tract and in the extreme case, the narrowing may result in total obstruction. Sounds involving such total obstruction come under the general heading of stops. Nasal consonants are produced when the nasal branch is open. Examples of this can be found in the final sounds of sim, sin and sing. Voiceless plosives are generated when the nasal branch is closed and pressure builds prior to the oral closure being release. Examples include the sounds at the beginnings of

the words pin, tin and kin.

Airflow through a constriction causes turbulence. When this process is steady, the resulting sound is know as a fricative, either voiceless (as in the initial sounds of fat, sip and thick) or voices (as in the corresponding sounds in vat, zip and the). The noise-like component of voiced fricatives is generally much weaker than that in their voiceless homologues. Indeed, the whole sound is less intense, and this intensity difference forms one of the cues used in discriminating between voiced and voiceless fricatives.

When the vocal tract is narrowed but not enough to cause turbulence a class of consonant sounds such as the initial sounds in way, ray and lay is produced. They are lumped together under the general heading of sonorants.

Whether the excitation in a speech sound is voiced or voiceless, the acoustic signal generated by the excitation is modified by the resonant structure of the vocal tract, which behaves as an acoustic tube along which planar propagation of sound waves occurs. Differences in the cross-sectional area along the length of the tube cause reflections, and it is these reflections that give rise to the resonances or *formants*. The resonant structure therefore depends on the position of the tongue, lips and jaw.

The generation of the excitation and its spectral modification by the vocal tract turn out to be largely independent of each other. To a good approximation, they can therefore be considered as a source isolated from, and leading into, a linear filter. If we look at the power spectrum of a section of speech waveform, the regularly spaced spikes occur at each integer multiple of the fundamental frequency of the excitation, and are harmonics of the fundamental. The intensity of the harmonics is determined by the product of two factors. The first is the spectrum resulting from the details of the airflow through the larynx from one closure of the vocal cords to the next; and the second is the spectrum corresponding to the impulse response of the vocal tract.

The frequency sensitivity of the ear can be measured in various ways – by having listeners determine subjectively equal frequency intervals at different locations in the spectrum;

by testing ther ability to detect small changes in frequency; by measuring the frequency range over which spectral componentns interact; or even by direct physiological measurements on the inner ear. All these methods lead to strikingly similar perceptual frequency scale, with sensitivity being roughly constant over the first few hundred Hertz and then decreasing with increasing frequency. The perceptual frequency scale is often approximated by a scale, the *technical mel scale*, that is linear to 1 kHz and logarithmic from then on [35].

Just as one might expect from signal processing considerations, the degradation in frequency resolution at higher frequencies is associated with an improvement in temporal resolution. This trade-off is well matched to the acoustic properties of speech. The higher formants have large bandwidths and do not therefore require high frequency resolution. In voiceless sounds, energy tends to be concentrated at high frequencies, representing large transients temporally. Spectral fine structure is absent, but such sounds, particularly voiceless plosives, contain events that are sharply defined temporally. Voiced sounds therefore require good frequency resolution at low frequencies (below 2kHz) and voiceless sounds require good temporal resolution above 2kHz.

Unless two frequency components are within a certain critical distance of each other on a perceptual frequency scale, their phase relationship has no perceptual effect. Consequently, a sound can be substantially characterized by its power spectrum, ignoring its phase spectrum. This characteristic will be exploited throughout this work.

Strong frequency componenets can suppress the ear's response to weaker components. In *temporal masking*, the strong component masks a weaker component at the same or a nearby frequency. The stronger component can occur just before or just after the weaker component, though the effect operates over much greater temporal separations in the former case – so-called *forward masking* – than in the latter. In simultaneous masking or frequency masking a strong component masks the presence of a weaker component presented at the

same time at a different frequency. The effect decreases as the frequency separation between the components increases, but the decrease is slower when the weaker component lies above rather than below the stronger component. Frequency masking therefore operates primarily upwards in frequency.

The perception of a continuous sound, such as a note from a musical instrument, is often divided into three parts: loudness, pitch, and timbre (pronounced ”timber”). Loudness is a measure of sound wave intensity, as previously described. Pitch is the frequency of the fundamental component in the sound, that is, the frequency with which the waveform repeats itself. While there are subtle effects in both these perceptions, they are a straightforward match with easily characterized physical quantities.

Timbre is more complicated, being determined by the harmonic content of the signal. Because of this, two waveforms can have very different time domain waveforms, yet sound identical. This is because hearing is based on the amplitude of the frequencies, and is very insensitive to their phase. The shape of the time domain waveform is only indirectly related to hearing, and usually not considered in audio systems.

One benefit to the ear being phase insensitive can be understood by examining how sound propagates through the environment. Suppose you are listening to a person speaking across a small room. Much of the sound reaching your ears is reflected from the walls, ceiling and floor. Since sound propagation depends on frequency (such as: attenuation, reflection, and resonance), different frequencies will reach your ear through different paths. This means that the relative phase of each frequency will change as you move about the room. Since the ear disregards these phase variations, you perceive the voice as unchanging as you move position. From a physics standpoint, the phase of an audio signal becomes randomized as it propagates through a complex environment. Put another way, the ear is insensitive to phase because it contains little useful information.

However, it cannot be said that the ear is completely deaf to the phase. This is because a phase change can rearrange the time sequence of an audio signal. An example is the

chirp system that changes an impulse into a much longer duration signal. Although they differ only in their phase, the ear can distinguish between the two sounds because of their difference in duration. For the most part, this is just a curiosity, not something that happens in the normal listening environment.

It is often said that timbre is determined by the shape of the waveform. This is true, but slightly misleading. The perception of timbre results from the ear detecting harmonics. While harmonic content is determined by the shape of the waveform, the insensitivity of the ear to phase makes the relationship very one-sided. That is, a particular waveform will have only one timbre, while a particular timbre has an infinite number of possible waveforms.

The ear is very accustomed to hearing a fundamental plus harmonics. If a listener is presented with the combination of a 1 kHz and 3 kHz sine wave, they will report that it sounds natural and pleasant. If sine waves of 1 kHz and 3.1 kHz are used, it will sound objectionable.

This is the basis of the standard musical scale, as illustrated by the piano keyboard. Striking the farthest left key on the piano produces a fundamental frequency of 27.5 hertz, plus harmonics at 55, 110, 220, 440, 880 hertz, etc. (there are also harmonics between these frequencies, but they aren't important for this discussion). These harmonics correspond to the fundamental frequency produced by other keys on the keyboard. Specifically, every seventh white key is a harmonic of the far left key. That is, the eighth key from the left has a fundamental frequency of 55 hertz, the 15th key has a fundamental frequency of 110 hertz, etc. Being harmonics of each other, these keys sound similar when played, and are harmonious when played in unison. For this reason, they are all called the note, A. In this same manner, the white key immediate right of each A is called a B, and they are all harmonics of each other. This pattern repeats for the seven notes: A, B, C, D, E, F, and G.

Speech is a non-stationary signal while pitch determination looks at the "simple" task of extracting fundamental frequency and period from such a signal. This is inherently difficult task because of: 1. the non-stationarity of speech; 2. the variation in articulatory gestures

leading to a wide range of possible temporal structures; 3. the variance of fundamental frequency, almost four octaves; 4. irregularity in the excitation signal.

Pitch and voicing parameters are different from analyses that look at vocal tract parameters. Within the scope of this research, parameters associated with pitch were used as the processing basis. Pitch, i.e., fundamental frequency (or rate of vocal fold vibration) F_0, as well as fundamental period T_0, has a key position in the acoustic speech signal. Pitch determination deals with voice source analysis. The parameters that are being determined are the manner of excitation (the presence of a voiced or voiceless excitation) and the rate of vocal cord vibration (referring to the pitch or fundamental frequency). The ear is an order of magnitude more sensitive to changes of fundamental frequency than to changes of other speech signal parameters [33]. Pitch or fundamental frequency is the predominant factor in the prosodic information of the speech signal. Considering the ear sensitivity and prosodic information, pitch determination is a key to good and reliable speech measurement methods.

Pitch detection algorithms can be categorized into Short-Term and Time-Domain analysis algorithms. With completely stationary and periodic signals, these algorithms yield similar results. However, variables such as stationarity/time variance, windowing length, averaging, and operating domain will influence the results of individual algorithms. Within the short-term algorithms there are many methods of pitch detection including: Correlation techniques such as autocorrelation and average magnitude distance function, Maximum likelihood techniques, and Frequency domain analysis techniques such as Harmonic analysis and Cepstrum analysis. Cepstrum analysis is of particular interest because the overall architecture lends itself well to a parallel analog implementation.

1.3 Existing Approaches in Analog/Digital Speech Recognition

Speech recognition systems have been implemented using a wide range of signal processing techniques including Neuromorphic/biologically inspired and Digital Signal Processing

(DSP) techniques. Neuromorphic/biologically inspired techniques such as silicon cochleas [44, 45, 47, 85] and Neural Network implementations [72, 66, 62, 46, 82] are based on simple and highly parallel computation and computational units. While DSP techniques are based on block transforms and statistical or error minimization methods that occur serially [65, 9, 74, 16, 86, 87]. In addition to this, the input signal must first be converted from analog to digital before the computation.

Over the past years DSP has grown into a very popular topic. Applications using DSPs have also grown, fuelled by the increase in computing power and decrease in cost, which has been a very positive step, particularly in the area of speech recognition. Analog signal processing chips that are able to perform the initial stages of speech recognition before passing the data to a digital processor would offer several advantages over conventional systems in power dissipation and higher speed (closer to real-time implementation).

With all that being said, algorithms aside, the different techniques still leave the huge question of implementation. Historically, many computational systems began as analog implementations and as digital systems became cheaper, faster, and demonstrated their benefits from ease of programmability (reducing design time), much of current signal processing has migrated to digital signal processing implementations. Going one step beyond this, digital systems have powered a significant portion of the advancement in signal processing applications. The current paradigm is to perform as much processing as reasonably possible, within the design constraints, in the digital domain for the benefits listed above. Neuromorphic implementations cover all analog [54] to all digital FPGA implementations [84]. So even in the implementation, the range is very broad.

Speech recognition is typically handled in two steps, Analog-to-Digital conversion followed by large amounts of computation performed on one or more digital signal processors. This approach minimizes the time the input signal spends in the analog domain and is primarily due to the improved processing efficiency of digital signal processing as compared to most analog signal processing circuits. In this case, processing efficiency is measured in

Millions of Multiply- Accumulates per milliWatt (MMAC / mW). An increase in processing efficiency equates to more MMACs / mW or same MMAC and less power. In addition to this, specialized Digital Signal Processors have made signal processing problems workable tasks and have proven to be a viable solution.

Specialized DSP architectures have been designed to increase computational efficiency while performing many of todays signal processing algorithms. DSPs, however, are very power hungry, and because of this, have not lent themselves to extensive processing tasks in portable devices. Furthermore, digital signal processors can require large amounts of die area for memory, which, at times, could be even larger than the processor itself. Typical applications are very scaled-down custom versions in order to meet specific power budgets. Power is directly related to the amount of computation being performed [81, 38], or for a DSP, the MIPS (Millions of Instructions Per Second) being executed.

Advances have come as geometries have decreased, following Moore's Law [14]. Figure 3 shows digital processing and the steady decrease in power per computation vs. time. Power and die area reductions can also be realized through custom ASICs designed to perform a specific processing task. One step further, mixed-signal and analog implementations show further reductions in power an area.

On the other hand, using biology as an example, low-power and robust analog signal processing is proven to be an efficient solution everyday in all living organisms. Biological approaches are typically massively parallel, highly noise tolerant and very low-power systems [57]. ASICs that are based on biological approaches are able to outperform general purpose processing systems. Taking this one step further, for a given task, where an analog implementation is possible, an analog implementation provides significant benefits in power consumption and processing speed, on the order of 2 to 3 orders of magnitude power reduction [70].

Neuromorphic approaches to speech recognition have focused on cochlea modelling and Neural Network approaches. This makes sense because of the biological attachment

to cochlea forms of processing and the inherent parallel processing of neural networks. Neuromorphic/biologically inspired techniques tend to fall into two categories a digital implementation of a biologically inspired circuit, or an analog implementation of the same. Rarely will you see a mix of the two with instances of processing very similar to digital signal processing. However, digital signal processing has proven to be the best method, aside from the human brain, to perform speech recognition tasks. What is needed is a system that combines the parallelism of neuromorphic circuits with the processing power of digital processing techniques, such as block transforms, etc.

Speech recognition involves a feature extraction step, followed by various recognition steps which can be implemented using vector matching to a stored set of feature vectors, Neural Network/Neuromorphic engineering, or statistical/probabilistic approaches such as Hidden-Markov Model decoding. Our system implements a continuous-time cepstrum as the feature extraction block. The cepstrum computation produces very useful features sets because it has an exponential frequency spacing for the filters. The cochlea has exponentially spaced frequencies which ties the cepstrum closely to biological processing for a well-suited balance.

Early models for speech recognition used straight forward feature vector matching. These were relatively simple to implement with algorithm differences primarily being changes to the error distance metric. Later approaches involved more Neural-Network architectures and Neuromorphic approaches took advantage of parallel processing and the underlying circuitry to improve recognition rates. Current approaches use statistical/probabilistic approaches such as building Hidden-Markov Models of speech signals. Hidden-Markov models have shown the most success in speech recognition systems with recognition rates above 90%. Each of the systems take feature vectors directly as inputs. We have combined the VQ and HMM approach because our architecture operates in the continuous-time domain, lending itself well to Neuromorphic and parallel architectures,

however the feature vectors are discrete pre-determined vectors, which simplifies the signal flow to later processing blocks. These pre-determined vectors are used to train and build our Pattern Recognition Block, which is based on Hidden-Markov Model decoding.

1.4 Neuromorphic Engineering

This research borrows heavily from IC design methodologies within the Neuromorphic community. Neuromorphic Engineering looks at doing quality engineering from a biological viewpoint. The responsibility for the circuits community is doing good engineering in parallel with developing systems that borrow from biological systems. Many of the designs have biologically inspired architectures and are used to perform problems/find solutions to tasks of a part of the body such as the cochlea [84, 69, 37, 59, 54], retina [56, 6, 3, 4], and at a lower level, systems of multi-neuron oscillators [60] down to single neurons [55, 63, 18]. These families of Neuromorphic chips will serve as a model for our Analog Signal Processing (ASP) circuits because they are well understood, biologically inspired and integrate well with our standard computational memory technology.

The cochlea and human auditory system is a relatively simple and well studied system when compared to other systems such as vision and cortex. However, the human auditory system still represents a significant level of complexity when compared to modern computer systems. Early research in cochlea design focused on designing systems by modelling the cochlea [53] and higher levels of the auditory nervous system [85]. The cochlea itself is a complex three-dimensional fluid-dynamic system and implementations of it's function have involved low-pass/band-pass filter implementations [17] while other implementations have looked at the core mechanics involved in it's functionality [84].

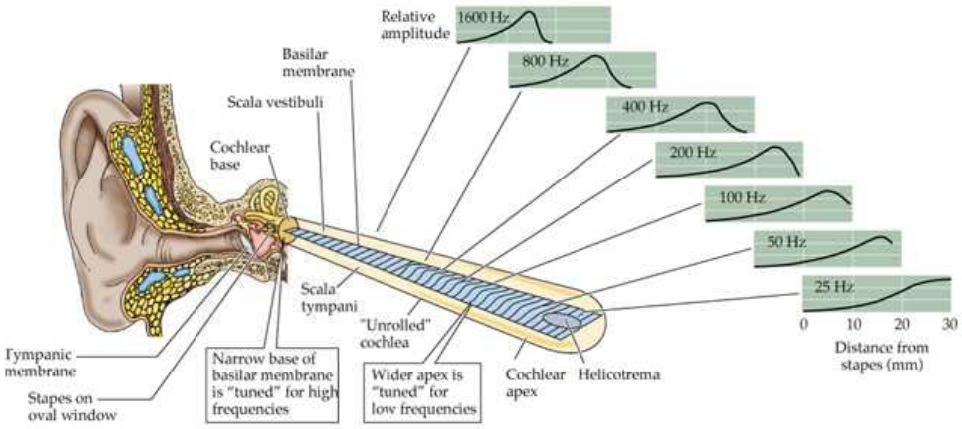


Figure 6. Frequency vs. position response of the cochlea. The cochlea responds to varying frequencies stronger at different positions. This system is similar to a filter bank decomposition.

1.5 An Analog Front End for Speech Recognition

The tendency in the signal processing realm for dealing with incoming audio signals has gone towards immediately passing the audio signal, which is in the form of an analog signal, to an analog-to-digital converter (ADC) so that the signal can be manipulated digitally. Typically, the FFT of the signal is performed digitally so that the individual subbands can be manipulated. Digital signal processing has many advantages, and the greatest is the ease of programming a digital system to meet the given requirements.

However, there is another option which is to introduce an analog system that does more than simply convert a signal into a digital version as soon as possible. By placing an analog signal processing block immediately before an ADC, as is shown in Figure 1, much of the processing can be done with the low-power and real-time computation of analog circuitry. This, therefore, alleviates a large portion of the burden of the digital circuitry. The overall system can either have a smaller digital processing block than was previously required, or it can have the same size digital block that will allow for more functionality since the basic processing has already been conducted in analog.

The final goal is to develop an An analog architecture for speech recognition that can work in concert with digital systems to offload some of the computation and allow the

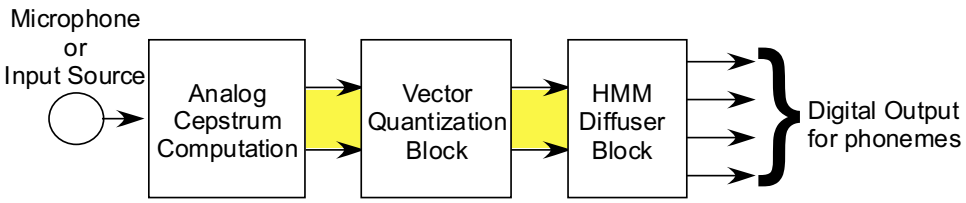


Figure 7. Analog speech recognition system block diagram.

DSP to perform higher-level more complex computations. The various processing blocks are basic blocks modified from ideal DSP blocks that are used throughout speech recognition systems. These include an analog Cesptrum-like processor that consists of an analog frequency decomposition similar to a fourier transform [77], a Vector-Quantization stage [29], that reduces the overall data set for later matching, and a continuous-time HMM block built from programmable diffusers [31]. The processing cores take advantage of the inherent computational abilities of analog circuitry and exhibit low power dissipation. Furthermore, by using floating-gate technology, each component block can be programmed to both eliminate any offsets and also reconfigure the overall system.

The architecture in Figure 7 serves as our starting point, as it was the initial block-level description of an analog speech recognition architecture [75]. Yet, in order to build a full system requires a through understanding of specific areas including: large-scale programming, filter banks, peak detectors, linear transconductance elements, vector matrix multipliers, vector quantizers, and programmable diffuser elements. Large-scale programming will be covered in the next chapter, followed by a discussion of filter banks and the design considerations associated with their use. The next chapter will cover the components necessary for analog signal processing operations, followed by a chapter discussing the analog signal processing systems that can be developed using these components including cepstrum processing, vector quantization, and programmable diffuser elements, all of which fall under the scope of creating a useful analog architecture for auditory feature extraction and recognition.

We commonly use several basic circuit elements for our auditory signal processing

structures, Figure 71 on page 99 shows one example system using these circuits. We will look at these circuits, as well as others, in the following chapters. Floating-gate circuit techniques enable using these circuits for a wide range of signal processing functions [28].

1.5.1 Frequency Decomposition

We have been using coupled bandpass IC filter models for cochlear modeling, which are designed to be used for front-end signal processing. The spectrum decomposition is done using differential C^4 second-order-section bandpass filters [17]. For simplicity only one half of the differential structure is shown in Fig. 71a on page 99. The spacing of the bandpass filters is arbitrary because each can be programmed to have a desired high-frequency corner and low-frequency corner [27]. Programming the C^4s is handled as if each filter were two floating-gate elements [41].

As a bandpass filter array, the C^4 filter banks are not cascaded as are many cochlea models [57], therefore eliminating the typical distortion or noise accumulation. In speech, particularly in noisy environments, the signal power is more evenly distributed across a broad frequency range than a simple tone, and therefore allowing for large input amplitudes with minimal output distortion (higher system signal-to-noise ratio). As a result, we typically have signal amplitudes through each filter that are 10mV to 30mV or less for input amplitudes between 0.25V and 1V, resulting in harmonic distortion through the system less than -30dB at each tap; differential circuits will further reduce these effects.

1.5.2 Amplitude Detection

The output of each filter bank passes through a peak detector stage to produce a constant magnitude output, or an envelope of the incoming signal. This magnitude is similar to taking the power spectrum density or real spectrum of an input signal. The circuit is shown in Figure 71b on page 99. We program the peak detectors to the desired frequency response of each frequency band. The floating-gate transistor on the output provides an offset current to set the DC output voltage. Each peak detector has an individually programmable corner

frequency. Because the output magnitude is continuous, this allows us to capture additional high frequency content within each band. The peak detector programming blocks are isolated similarly to the C^4s. The entire bank is treated as a single row and within that row the individual elements are accessed by column. Control circuitry on the rows and columns ensures isolation.

1.5.3 V-to-I Linear Transconductor

The output from each amplitude detector block is a voltage signal. In order to generate the log of this voltage, which represents the magnitude, we need to transform this signal from the voltage to current domain. The major hurdle is doing this operation linearly over a wide-enough range for the signal to be useful. This section gives an initial circuit description for such a block.

1.5.4 Weighted multiplication

Figure 71 on page 99 shows our analog differential multiplier that multiplies the incoming differential voltage signal with a stored differential weight. We program the positive and negative weights by setting programmable floating-gate voltages. These values can be programmed to any arbitrary value, Their differential operation requires each pair to have a DC bias voltage.

1.5.5 Distance Measure

In order to reduce the data set output from the continuous-time cepstrum operation, we will quantize each vector to a know vector stored in memory. As a parallel operation, we are able to perform the quantization stage using an array of single distance measure blocks implemented using ”bump” circuits. This provides a programmable, parallel and programmable method of performing large sets of vector distance measures instantaneously.

1.5.6 Sequence Detection

The final stage of recognition will involve a sequence detection step in order to match a set of vectors to a model for a given sound that corresponds to a phoneme, or sub-phoneme. At this stage the outputs are digital and the overall system lends itself well to cascading multiple stages together to produce higher levels of recognition.

With this as the basis, the following chapters will look at each of the required pieces in detail and outline the process toward developing an analog IC for auditory feature extraction and recognition. The chapters will begin with the programmable analog elements as they are critical to all of the following blocks and require quite a bit of discussion on their own. Following the programmable elements, I will cover the programmable filter banks, the first block in the system, and the most critical block for the feature extraction phase. The next chapter will discuss all of the ancillary blocks required, including those needed to maintain a useable signal flow from the filter banks through each of the core blocks used in each of the following recognition blocks. The final chapter will discuss the recognition blocks of the system in detail, with the final goal of this thesis being the introduction of an analog architecture for auditory feature extraction and recognition.

CHAPTER 2

PROGRAMMABLE ANALOG MEMORIES

This chapter will discuss the core memory element used throughout the analog computing arrays, the floating-gate transistor. The floating-gate transistor is a standard p-type transistor with no DC path to ground leading from the input gate of the device. This allows for charge, and a resulting effective voltage, to be stored on the gate with negligible degradation in the stored value. Critical characteristics of the device making them useful to us include: it's ease of integration into standard analog circuit designs; the combination of memory within the computational element, reducing area; and the accuracy of values stored in the element.

2.1 Floating-gate Transistor Element

The elements used in the computational memory structures are vanilla pFET transistors with additional polysilicon area to accommodate a poly-poly capacitor as the input to the device, as shown in Figure 8. This structure provides a method for storage and computation within the device.

The use of floating-gate devices as a memory element is not new in the circuits community [10]. Since their discovery much effort has gone into making them a viable non-volatile memory element which we now find in digital cameras and mp3 players. More recently, the beneficial use of floating-gates in analog circuits has been realized and it has been published that not only can they be used as memory devices but function as programmable, compact, computational elements [39, 19, 23].

Floating-gate technology is used in Electrically Erasable and Programmable Read-Only Memory (EEPROM) which use special processes. Floating-gate devices can also be created using standard CMOS processes which contain oxide thicknesses that are uniform over all active devices and on the order of $450 - 500\mathring{A}(1X10^{-10}m)$. Programmability is possible

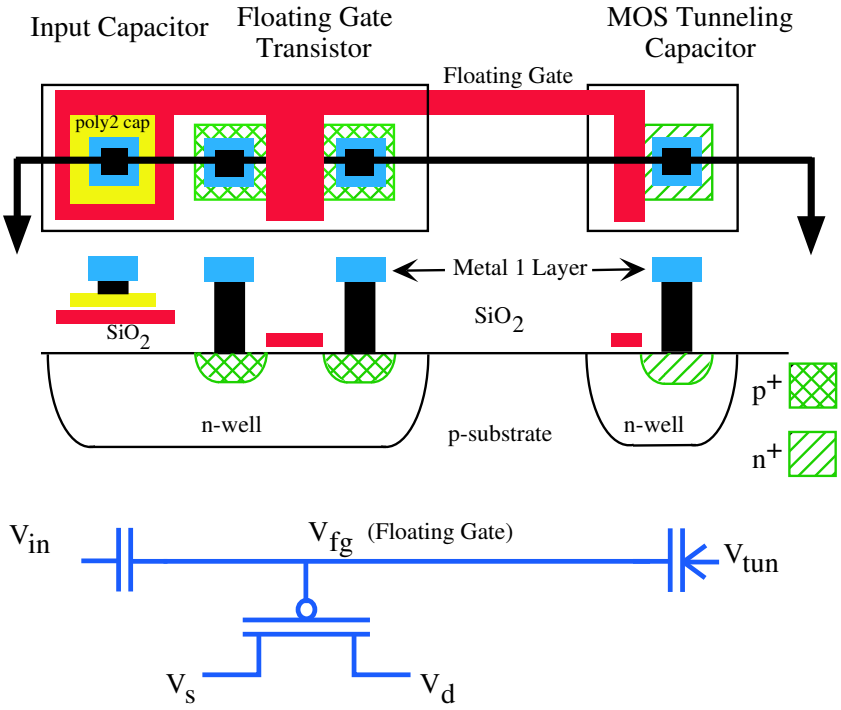


Figure 8. Cross section of a floating-gate element. The structure is very similar to a classic pFET transistor. The floating-gate is made from polysilicon (poly1), however, it does not have any physical contacts to it's surface and is electrically insulated by silicon-dioxide, thus able to store charge that is on the gate indefinitely. The poly1 gate can be modulated by the poly-poly capacitor that is formed by eletcrolyte (poly2) and an input gate voltage (V_{in}). Source-to-Drain current is modulated through the capacitor divider from the gate input into the channel through the series capacitance from gate to channel. Tunnelling occurs through the MOS capacitor that is formed from the polysilicon gate to the nwell. The nwell voltage (V_{tun}) is controlled externally. Injection is controlled by modulating the channel current, by modulating the gate input voltage; and the drain-to-channel voltage, which is modulated directly through controlling the drain voltage. Programming is possible through the control of the injection and tunnelling phenomena.

using Fowler/Nordheim Tunneling [50] (Tunneling) and Channel-Hot Electron Injection [20] (Injection). Table 1 lists voltages required for the tunneling and injection phenomena for various processes and as technologies advance, the device geometries decrease leading to lower programming voltages. Lifetime and variability of stored values on the floating-gate, in the listed geometries, is not an issue as shown in [11]. However, smaller oxide thicknesses and higher electric fields raise interesting design considerations with regard to retention as geometry sizes continue to decrease.

There are however, fundamental differences in usage between EEPROM, multi-level

memories and these computational memories in that the programming values do not represent discrete levels of charge being programmed to the device. This is both a limitation and benefit, depending on how the device will be used in subsequent circuitry. For digital circuits the desired solution is only interested in resolving two distinct values for each cell, one or zero Newer digital systems store information using multiple levels but fundamentally use approaches similar to binary valued cells. Conversely, an analog memory has infinite accuracy, theoretically limited to a single electron moving across the barrier and require a continuous range of programmable values. Additionally, instantaneous programming of each cell is not as critical because in an analog system we are generally more interested in programming a device once and then reading or, as we use them, computing continuously afterwards. This represents a fundamental trade-off between precision and time and thus, the programming scheme for analog cells must be different from that used for digital memory. Designing for large scale integration of these memory elements into systems requires that the elements themselves be relatively small, however they can be scaled and sized just as traditional transistors. Due to their small design sizes, mismatch between elements can be noticeable, as one would expect from any standard process. The ability to program each device, in addition to smart integration techniques such as programmable current mirrors and proper layout, we are able to show mismatch cancellation equivalent to 14-16 bits of accuracy [1]. Exploiting this accuracy allows us to program a single device down to the sub-picoammeter range, by including some considerations for the fundamental mechanics of the transistor.

2.2 Device Programming

As mentioned earlier, a key requirement for using floating-gate elements is the ability to accurately program each element. Programming the analog memory elements is a complicated task as outlined in [80, 42]. By controlling the various device parameters such as gate voltage and drain-source voltage we are able to accurately program the devices using

Min. Channel Length	Oxide Thickness	Tunnelling Voltage	Injection Voltage
$2.0\mu m$	$400 - 450\mathring{A}$	27 V	14 V
$1.5\mu m$	$300 - 350\mathring{A}$	20 V	10 V
$0.5\mu m$	$110 - 160\mathring{A}$	15 V	6 V
$0.35\mu m$	$70 - 85\mathring{A}$	12 V	5 V
$0.25\mu m$	$50 - 60\mathring{A}$	8 V	4 V

Table 1. Tunnelling and injection voltages for various feature sizes available through MOSIS (http://www.mosis.org). The required voltages start noticeably high for minimum channel lengths of $2.0\mu m$, but as the oxide thickness decreases, each of the programming voltages also decreases. Current programming voltages are now within the range of using on-chip charge pumps and high-voltage control circuitry. Tunnelling refers to Fowler/Nordheim tunnelling [50]. Injection refers to Channel Hot-Electron (CHE) injection [20].

Hot-Electron Injection and Fowler-Nordheim Tunnelling. Other mechanisms such as UV Photo injection may also be used, but are not used in our programming schemes due to process variations. Table 1 list various programming voltages over different processes. The required voltages start noticeably high, but as the oxide thickness decreases, each of the programming voltages also decreases. Current programming voltages are now within the range of using on-chip charge pumps and high-voltage control circuitry.

Using large floating-gate arrays on the order of 1K to 10K elements, it becomes obvious that programming each element by hand would be a very time intensive process. Figure 9 shows a block diagram of the programming algorithm. The programming algorithm has been automated in MATLAB and comprises three lower level functions. These functions include:

1. EraseArray

Tunnels entire array until all current levels are below their desired values. This function is required because tunnelling is used for global erase and injection is used for fine programming but only works in a single direction.

2. GetInRange()

Ensures that all elements have a sufficient current at a given gate voltage to increase

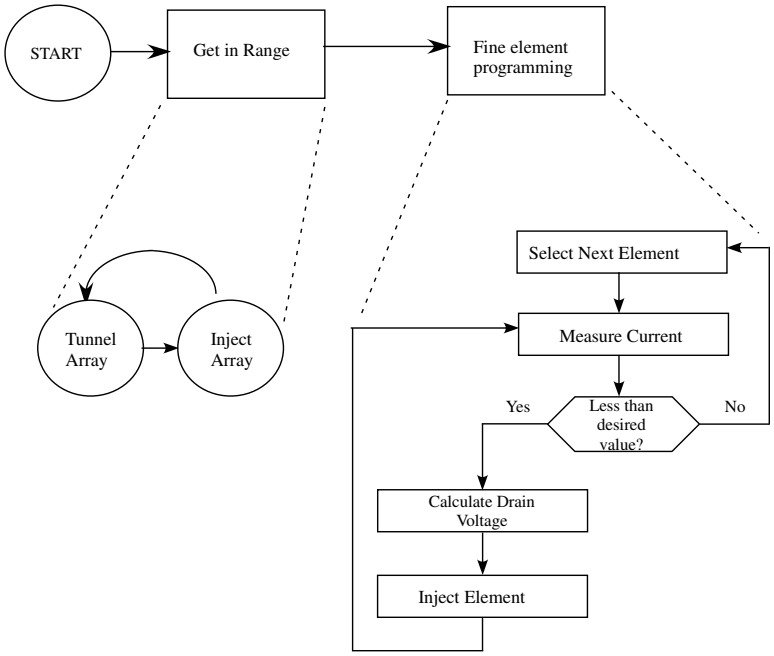


Figure 9. Flow chart of first iteration of floating-gate programming algorithm. Future revisions use similar relation for calculating the necessary injection voltage but use on chip circuitry to program multiple elements in parallel and also to measure currents on chip.

the speed of injection. Injection requires drain current. Tunnelling is a global function so some elements may have significantly lower current levels than others. Future revisions will allow isolation of tunnelling as well.

3. InjectArray()

Controls the injection of an entire array by calculating the optimal drain voltage to ensure that each injection pulse brings the element closer to the desired current without overshooting.

The programming algorithm is based upon a representation of the physical channel hot-electron (CHE) injection equation as shown below, and solving for the injection drain-to-source voltage required to move the device output characteristics to a desired value. This relation is calculated and iterated for each device to achieve a final output value of the memory element.

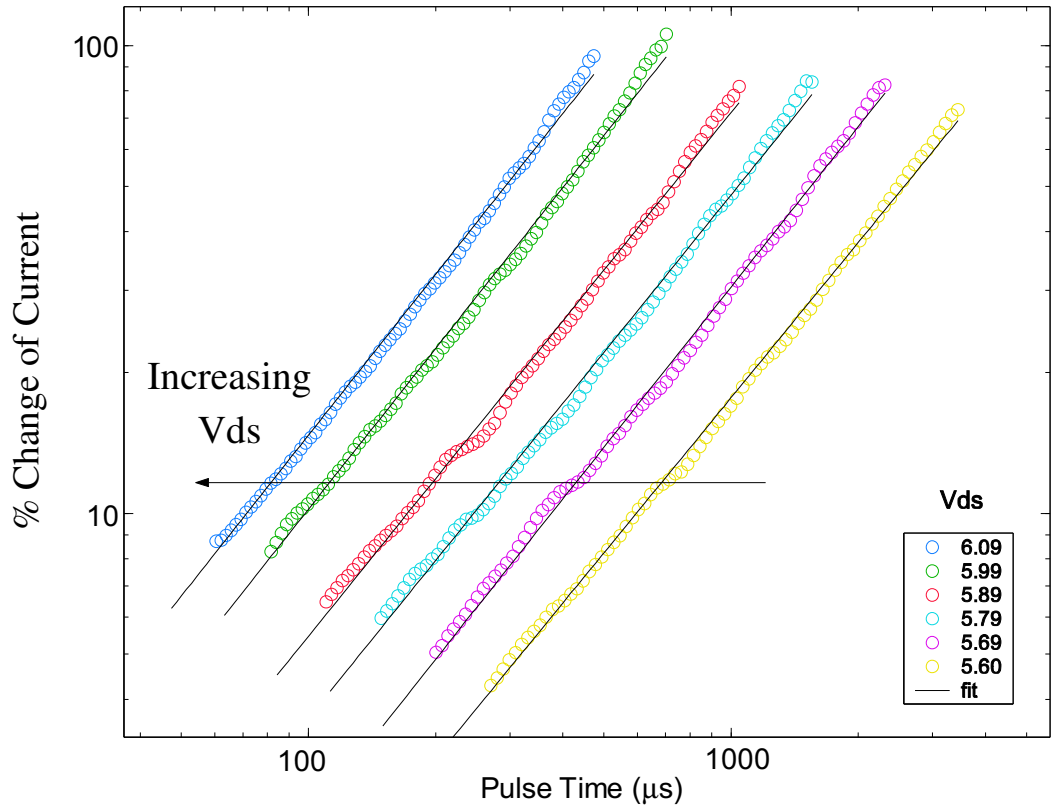


Figure 10. Plot of injection rate versus injection pulse width for different drain-to-source voltages. The injection pulse width was limited by programming hardware and is shown to occur at a pulse width of 100μs. The minimum injection pulse width can be decreased by moving the control circuitry on chip.

Solving for the injection drain-to-source voltage for a desired current using the Hot-electron injection in a pFET is described by (1)

$$I_{inj} = I_{inj_0}\left(\frac{I_S}{I_{S_0}}\right)^{\alpha} e^{-\Delta V_d/Vinj} \tag{1}$$

where I_{S_0} is the initial current, I_S is the final current, and V_{inj} and I_{inj_0} are physical device parameters. The injection current is also dependent upon the change in the floating-gate voltage V_{fg} [26]. Solving for I_S we get

$$I_S{}^{-\alpha} = \frac{-\alpha \cdot I_{inj_0}}{C_0 \cdot I_{S_0}{}^{\alpha}} \cdot e^{-\Delta V_d/V_{inj}} \cdot (t) + I_{S_0}{}^{-\alpha} \tag{2}$$

where $C_0 = U_T \cdot C_T/\kappa$ [22, 24].

Rearranging this solution into a single equation for ΔV_d, we get a solution that is the required drain voltage to reach a desired current. A change in drain voltage is calculated

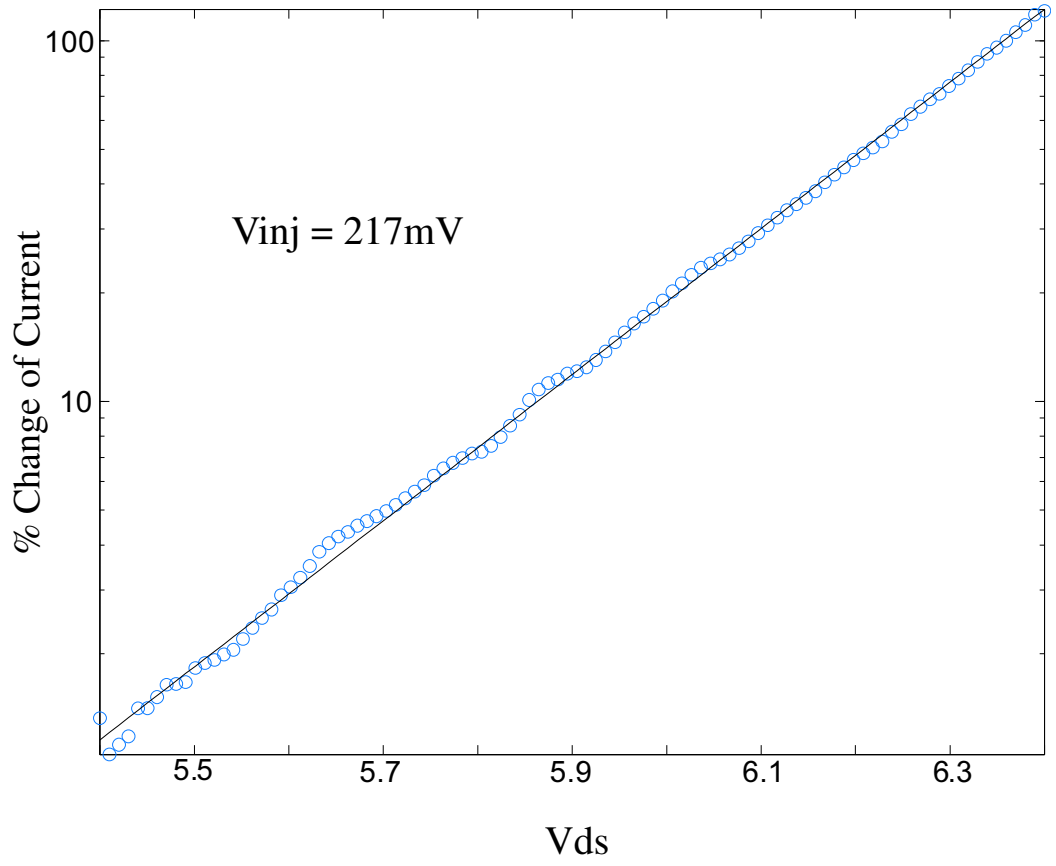


Figure 11. Using the injection rate parameters in Figure 10 we are able to extract the values for V_{inj}.

around a quiescent drain voltage that gives a 10 percent change in current for each injection pulse.

$$\Delta V_d = -ln\left[\frac{-C_0}{\alpha \cdot I_{inj_0} \cdot (t)}\left[\left(\frac{I_{S_0}}{I_{S_{desired}}}\right)^{\alpha} - 1\right]\right] \cdot V_{inj} \tag{3}$$

The required drain voltage is calculated for each element in the array given their present value of current and the final desired current. These voltage values are then applied to each element in the array.

Accurately programming a device is highly dependent upon the accurate representation of the device parameters. There are methods for extracting the parameters as will be explained shortly and each extraction is dependent upon the accuracy of the measurement equipment, which is where we see a direct relationship between speed and accuracy. The device parameters such as α and V_{inj} relate to physical parameters on the IC obtained

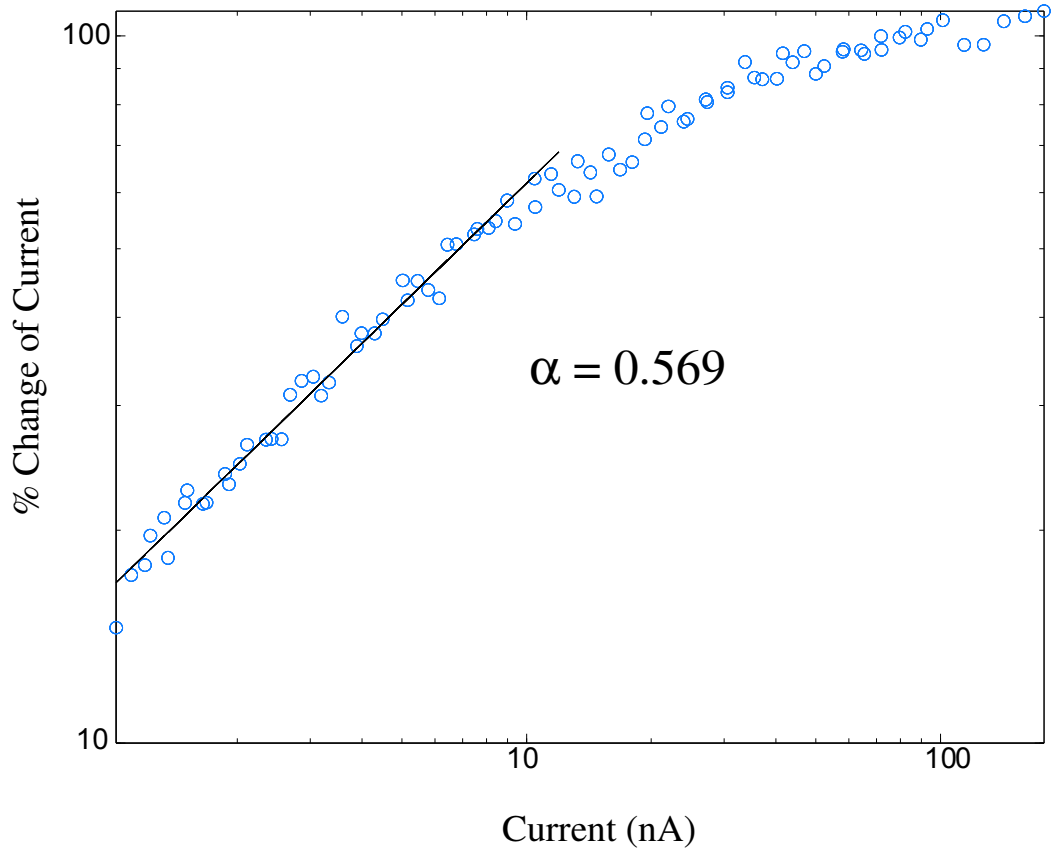


Figure 12. Using the injection rate parameters in Figure 10 we are able to extract the values for α.

through curve-fit extraction, where as I_{inj_0} is a constant.

Figures 10 shows the injection process at multiple values of drain-to-source voltage. The plots are typically done over multiple devices to speed up the process. This data shows ΔI on the Y-axis and I on the X-axis because we know injection is current dependent. From the injection rate equation

$$\Delta I = m \times I^{1+\alpha} exp(\frac{-\Delta V_{ds}}{V_{inj}}) \tag{4}$$

On a log-log scale,

$$\ln(\Delta I) = \ln(m) + (1+\alpha)\ln(I) + \frac{-\Delta Vds}{V_{inj}} \tag{5}$$

we see that the slope of each line equals $1 + \alpha$. Figure 12 shows the extracted value of α. Next, using Figure 10, V_{iinj} can be extracted as follows. Choose a current I to be the base current for operation. From the selected I, extract the vertical column of data from

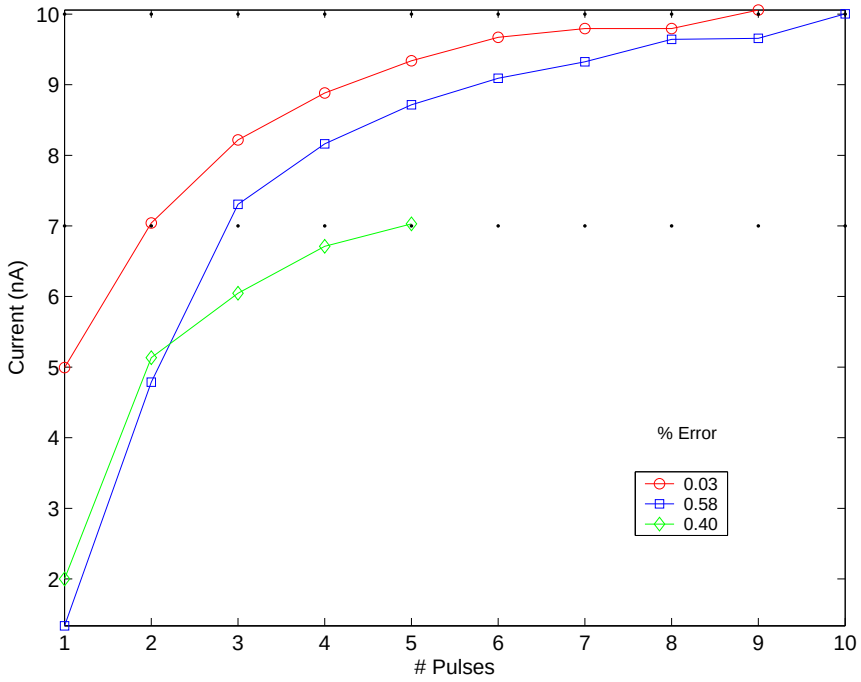


Figure 13. Example data using the programming algorithm to program a device to various operating points. This plot shows convergence of three different trials twice to 10nA within 10 steps and once to 7nA within 5 steps.

Figure 10. The resulting plot will resemble data in Figure 11 and the slope equals $\frac{I_{desired}}{V_{inj}}$. Performing a similar linear fit we are able to extract V_{inj}.

Injection has been shown to occur with pulses down to 10μs as shown in Figure 10. The rate of change at a given drain-to-source voltage increases with pulse width and drain-to-source voltage. Figure 13 shows convergence in eight to ten iterations, which results in an individual programming time of 80 - 100 μs. For an array with 1K floating-gate elements, the total programming time could be as low as 80 - 100 ms.

The speed of our programming algorithm can be estimated as follows:

$$T = \left[\left(t_{meas} + t_{pulse}\right)\cdot E + t_{ramp}\right]\cdot P \qquad (6)$$

where t_{meas} is the time required to perform a current measurement, t_{pulse} is the time required for a pulse, t_{ramp} is the time required in ramping, E is the number of elements to be injected, P is the number of pulses, and T is the total time required to inject E elements. Although

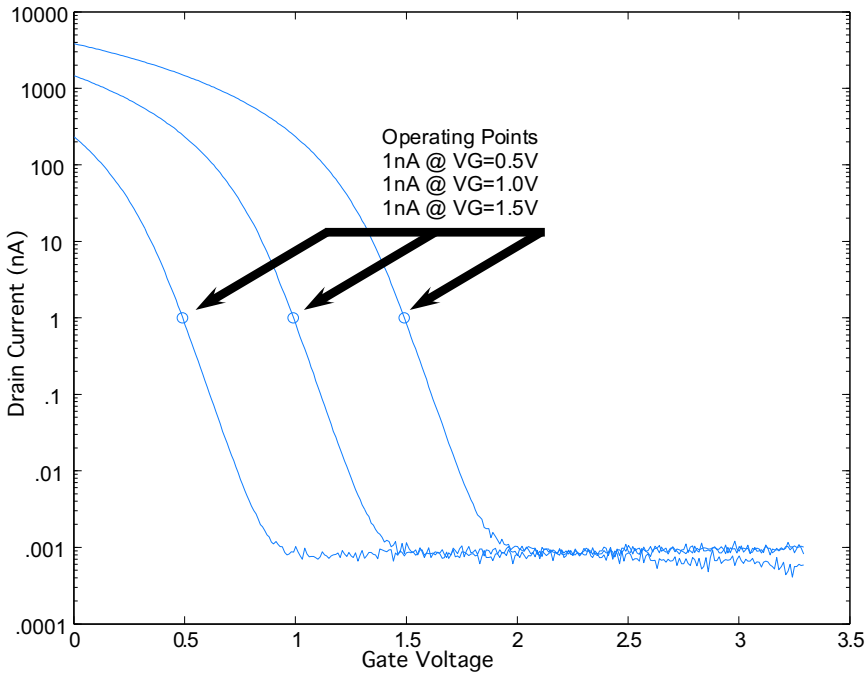


Figure 14. Floating-gate programmability of a single device to various operating points. An operating point is defined by a Drain Current at a given gate voltage. This figure shows three (3) device operating points. Each operating point is reached by programming the device.

t_{meas} will decrease as the current increases, and overestimate of the programming time can be obtained by using the slowest current to be measured. Typical values for t_{pulse} and t_{ramp} are 10μs and 200μs. Using the FPGA board as the direct interface to our commercial ammeter, current measurement time is on the order of 100ms. It can be seen clearly that the dominant factor in our speed is the current measurement. Future implementation will use on-chip integrators with small capacitance values to decrease the current measurement time, thus reducing t_{pulse}. A 100fF capacitance will give us a t_{pulse} of 400μs (including average). Assuming P=10 for every element, it will take at most 2s to program a 500 element array.

The programming algorithm is computational intensive and requires solving (3) before every injection pulse. Current implementations transform (3) into a look up table that is loaded into the FPGA, thus avoiding any kind of computation and greatly increasing the programming speed. Now the flow of data is only between the PC board and the FPGA.

Using this predictive algorithm, one is able to reliably and repeatably shift the operating point of a single memory element along a continuous range of values for gate vs. drain current as shown in Figure 14. Speed is still relatively important as large arrays are to be programmed, but our approach prioritizes accuracy considerations over speed as evident by the iterative approach.

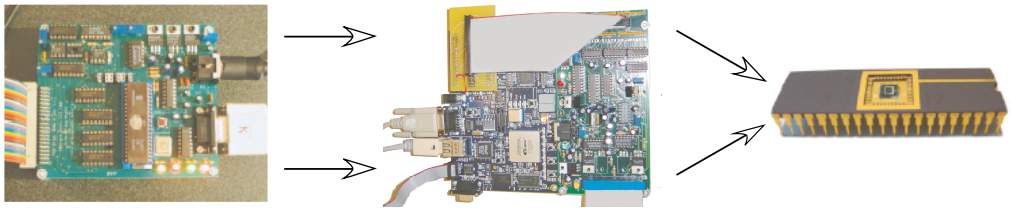
2.3 Programming Hardware

We designed a custom floating-gate programming board to control the floating-gate programming scheme. The board is controlled by a serial port through MATLAB. The board controls the power supply, gate voltage, tunnelling voltage, and drain voltage. The board is also able to measure current while varying the drain voltage, which is essential during injection.

Part of the system involved accessing the array is on chip with the devices. Future revisions of this system are being developed that move more of the external circuitry on chip. This will provide cleaner data and increase programming speed. Speed improvements are also possible by utilizing parallelism that can only be obtained, at reasonable costs of time, money, and overhead, by moving the control on chip.

Digital control of our PCB is done with the use of an FPGA. The FPGA on the programming board is an Altera Stratix EP1S10 device. This FPGA has approximately 200K equivalent system gates, just under 1 Mbit of on–chip memory, and 48 embedded multipliers. The Stratix FPGA is installed on one of Altera's standard development boards that contains 25 MBytes of external memory (including SRAM, SDRAM, and Flash), 10/100 Mbit Ethernet interface, and 81 general–purpose I/O pins.

The FPGA is configured to implement a customized soft–core processor (Altera's 32 bit Nios processor) along with specialized VHDL modules that handle timing critical communication between the programming board and the soft core processor. The Nios processor controls the overall system and coordinates the parallel operation of the different VHDL



(a) First Generation Programming Board (1G)

Programming algorithm implemented in MATLAB.
Slow Interface.
No on-board memory.
1 element / 10 min.

(b) Second Generation Programming Board (2G)

Programming algorithm implemented in VHDL.
High-speed interface
On-board memory.
100 elements / 1 sec.

(c) Third Generation Programming Board (3G)

Programming algorithm implemented ON-CHIP.
High-speed interface.
ON-CHIP memory/control
100K - 1M elements / 1 sec.

Figure 15. Programming control structure hardware evolution.

modules. In addition, software running on the Nios processor implements the TCP/IP protocol and communicates with the PC via a 100 Mbit Ethernet connection. On the PC side, a Matlab interface has been developed that provides a direct link to the FPGA from the Matlab command line.

Part of the system involved accessing the array is on chip with the devices. Future revisions of this system are being developed that move more of the external circuitry on chip. This will provide cleaner data and decrease programming speed. Speed improvements are also possible by utilizing parallelism that can only be obtained, at reasonable costs of time, money, and overhead, by moving the control on chip.

Programming board serial port communication is limited by port speed and also the operating system running the algorithm. Using windows machines has shown to add additional serial port timeout delays. We are currently moving the algorithm onto a programmable integrated circuit controller (PIC) or onto a field-programmable gate array (FPGA), thereby removing the OS delays.

A four layer PC board was fabricated. With the use of DACs, ADCs, and level shifters, this board provides the user with 7 bias voltages ($0 - 3.3V$), 4 programming voltages ($0 - 10V$), and 18 level shifted digital signals. It also allows for one analog voltage measurement

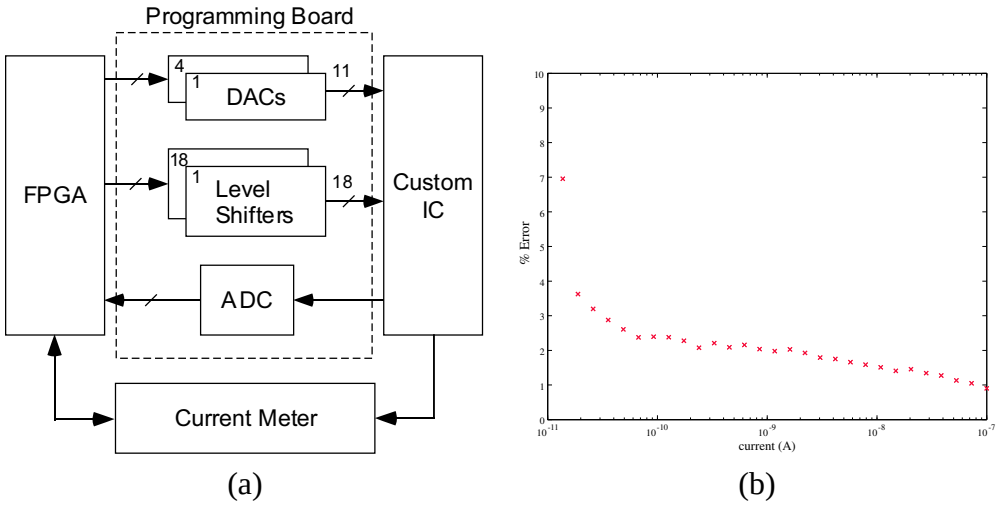


Figure 16. a.) Figure to illustrate the setup used to measure the device current and for the programming. Illustrate the ammeter, custom programming board and FPGA which implements a set of instructions from the programming algorithm. b.) The accuracy of the current measurement equipment decreases sharply as the current being measured decreased.

$(0 - 5V)$. A clock of 10MHz is used for the DACs, while the ADC has a maximum rate of 200M samples per second. Figure 16 (a) shows the block diagram for the board. The board is fully controlled with the use of an FPGA.

External current measurements are limited due to the huge line capacitance for wires running off chip and the equipment used to perform the current measurement. The current measurement circuitry can typically provide one current measurement between 10μs for large currents and 100ms for very small currents. Off chip also requires additional filtering of the data to ensure accurate results which can make the measurements even slower. Future versions will have on-chip current measurement circuitry.

The current measurements are the primary bottleneck on floating-gate programming. A precise and fast reading is required for accurate programming. Taking the current out of the chip into the ammetter introduces some error in our measurements. The noise and the parasitic capacitances introduced by the protoboard greatly decrease our accuracy for measuring the current. This will in fact affect our programming precision. Our current measurement accuracy is shown in Figure 16 (b).

Averaging will increase the accuracy of measurement but will also increase the time required for the current measurement. To minimize the time delay an "intelligent" averaging is performed. This consists on increasing the the averaging as we get closer to the target. This is implemented as a look up table that is loaded into the FPGA.

2.4 Kappa Projection Algorithm (KPA) for Ultra-low programming

Biological applications such as filtering for hearing-aids and signal decomposition in the auditory frequency range can have time constants as long as tens of milliseconds to seconds. Achieving such long time constants can be extremely challenging in integrated circuit technology due to limits in capacitance sizes. This becomes extremely obvious in Gm-C type analog filters where corner frequencies as low as 100Hz may be desired. For a corner frequency of 100Hz, τ would be approximately 1.5ms. Limiting capacitance to 1pF requires a Gm of 0.628nS corresponding to a bias current of 32pA, which for many, already represents a significant design challenge [51].

To make the problem even more challenging, for analog processing architectures such as Cepstrum Filtering [77], arrays of parallel analog filters (more than 10) are the initial processing step, making the use of 1pF capacitors for each filter unreasonable. Typical capacitance values for these filters are on the order of 50-100fF. Keeping the time-constants the same, but scaling currents, we are now required to provide bias currents from 100fA - 1pA for the low frequencies. This trend can be seen in Figure 21. Additionally, we need to be able to provide a large number of these bias values and also the ability to adjust each bias. This would allow the programming scheme to account for device mismatch and adjust for errors that cause a shift in the desired pass band of the filter.

These types of biasing schemes are particularly useful in low-power applications where the supply voltages are severely reduced. These elements allow us to program the device to any effective Threshold Voltage we desire. This programming method is particularly useful in a low VT process because vg can be programmed above / below gnd. Additionally, there

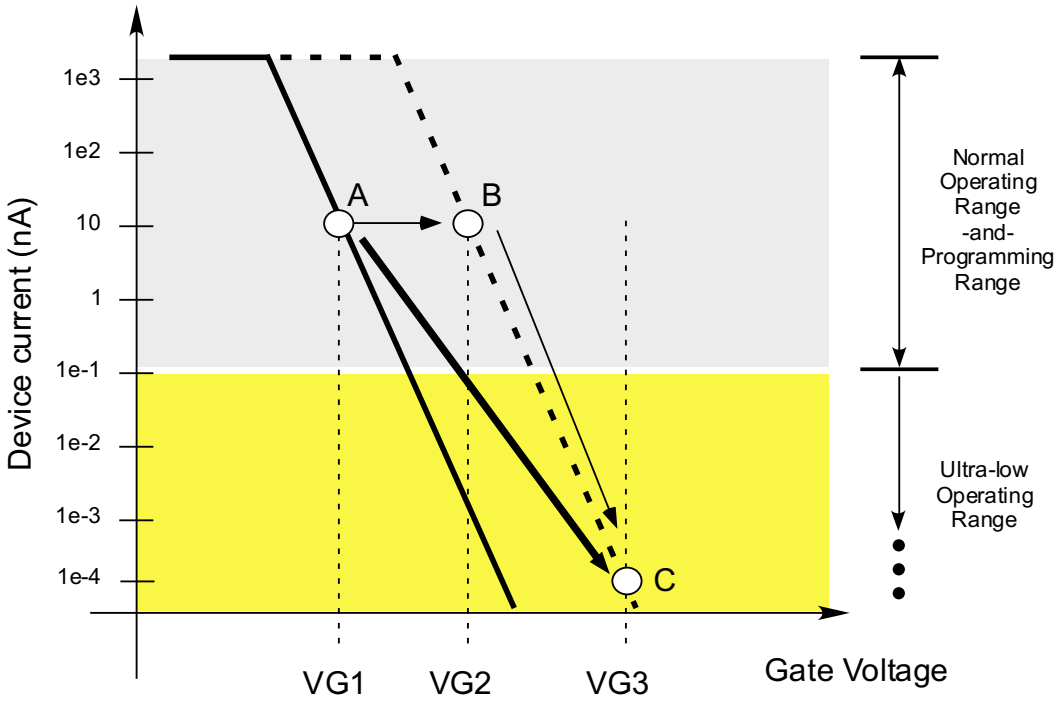


Figure 17. A novel predictive algorithm for programming floating-gate devices into pico-amp and sub-pico-amp current ranges. The desired operating point of the device is "C". "A" is the current operating point of the device. Rather than program the device directly to "C", the device is programmed to "B", taking advantage of the improved accuracy in that range for the current measurement equipment. Operating point "C" is achieved by taking into account the Sub VT slope of the device.

are no issues using this device as part of a current mirror to generate a current sink through an nFET.

The Kappa Projection Algorithm (KPA) is a method of predicting the final operating current of a device from a know, but higher starting current. The device parameter κ is critical to the deep sub-threshold current projection and accurately predicting the final operating point. Because the predictive algorithm uses this value to scale the effective shift, any error has a multiplicative affect on the final current. The κ of the floating-gate devices is decreased due to the capacitive coupling of the input as shown in Figure 18(a). Typical values for kappa $\sim$ 0.4 - 0.7 for a corresponding Cin $\sim$ 30fF. To make the problem more difficult, κ varies with current. Our extraction method fits κ over the entire sub-threshold range and uses the largest value. However, for much lower currents we predict that κ will

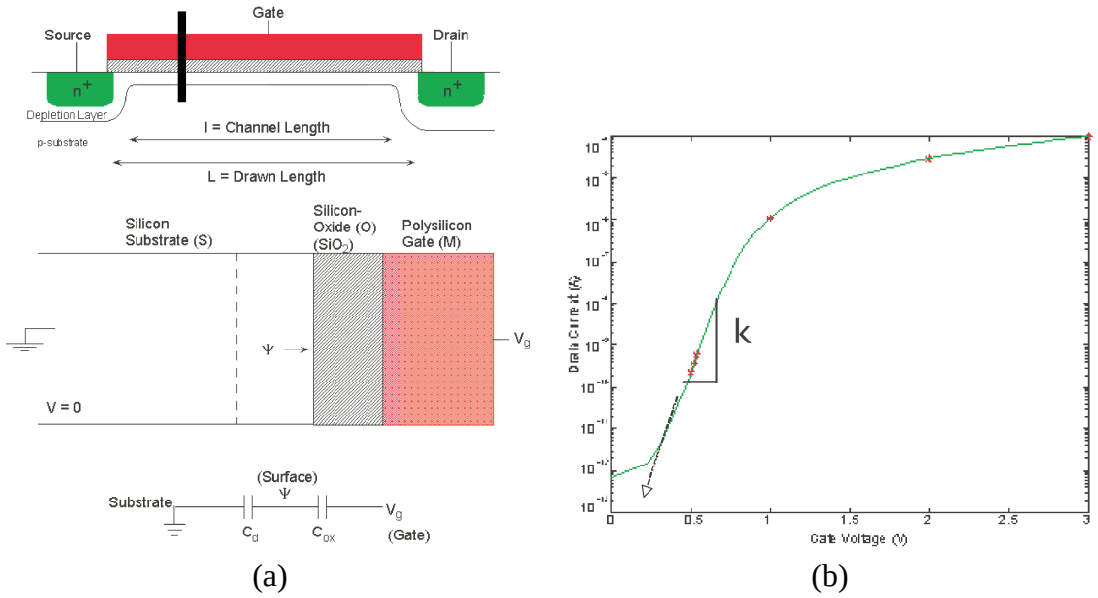
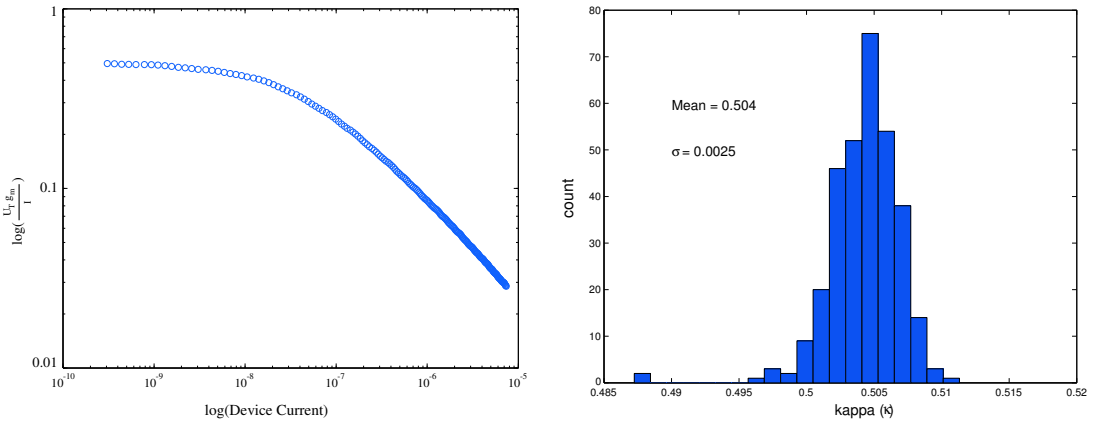


Figure 18. a.) Figure showing kappa and what is meant by the ”effective” kappa of a floating-gate device. Kappa represents the attenuation on the input gate voltage due to the capacitive voltage division from input to the channel of the device. b.) Using this effective kappa, we are able to project the operating current for a device well below the limits of our measurement equipment.

decrease introducing an error in the prediction. Figure 19 (a). shows how κ varies throughout the sub-threshold region.

The κ of a transistor relates directly to the amount of coupling achieved from the gate into the channel of the device. Floating gate devices use capacitively coupling into the gate of a transistor, thus the effective coupling from gate to channel is decreased due to a capacitive division from two series capacitors. There are two widely used methods for extracting this value κ from a single device. These include: direct extraction from the subthreshold slope of a device in saturation or extraction from the gain of a device in common-gate configuration. Figure 18(b) models the first of these two methods, which is what was used here.

Accurate measurements of current and voltage, respectively, is critical in each of these methods. Current measurement error using commercial current measurement equipment vs. on-chip integrator style current measurement has been investigated elsewhere [73]. For all the tests outlined in this chapter, commercial current meters were used. A sample data

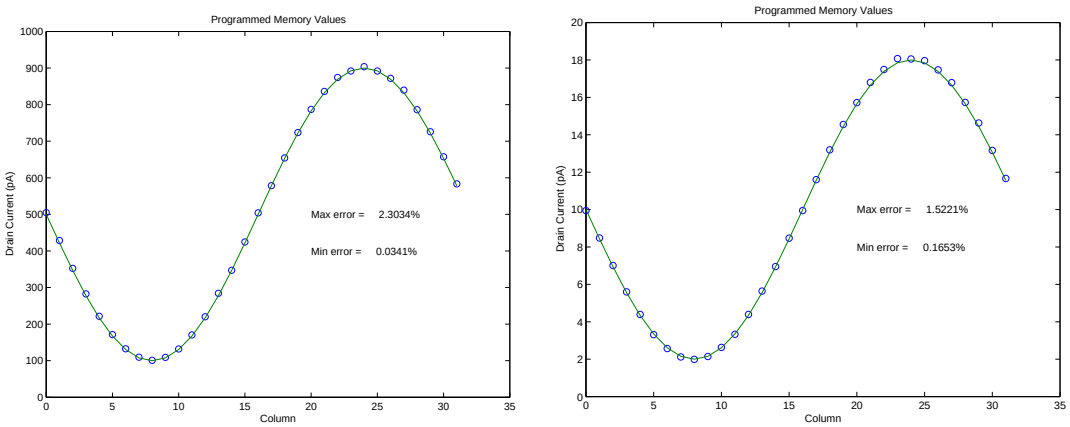


(a) (b)

Figure 19. κ variation with drain current and variation across an entire chip at the single deep sub-threshold projection point. a.) Average κ across the entire array of devices. κ changes with device current. The expression for κ is only valid in the sub-threshold region of operation. As the device current continues to decrease, we see that κ approaches an assymptote, but tends to continue to increase slightly. The projection data was generated using the largest value of κ measured, but this is still less than the actual κ for very small currents. This accounts for the error increasing as we project the operating point of the device to lower and lower currents. b.) Effective kappa variance across a chip of computational memory elements. $\kappa_{eff} = \kappa\frac{C_{input}}{C_T}$, where C_{input} is the input capacitance for the device and C_T is the total capacitance for the node. Variance in both X and Y direction is less than 1%.

set was taken for an array of 320 programmable memory elements. The κ std is 0.0025 as shown in Figure 19 (b). Thus all of our parameter extraction and predictive calculations depend on taking accurate current measurements to give an exact location for the current element. Current accuracy given our measurement setup using commercial picommeters, have given us an accuracy of 2% or less, within our desired range of operation as shown in Figure 16.

Other methods of current measurement include on-chip method such as integrating structures. These structures provide cleaner current measurements for low currents and when integrated with external digital control circuitry, are able to measure currents as fast as 100 - 1000 measurements per second. Noise in the system is included in the power supply noise along with other noise sources such as substrate coupling into the floating-gate, but these are well understood noise sources. The limitations for the on-chip current measurements are determined by the linearity dynamic range of the operational amplifiers used



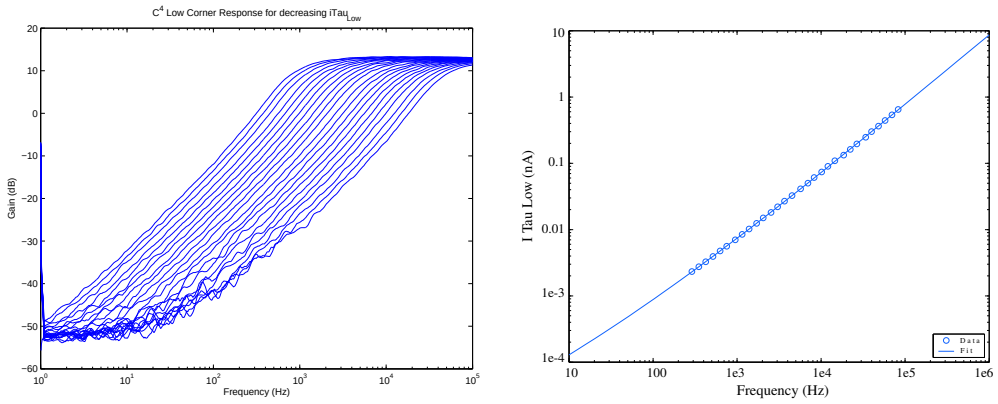
(a) (b)

Figure 20. Sub-threshold plots showing a set of programming points using the Kappa Projection Algorithm (KPA). Programmed currents were set as low as 2pA with a maximum error of 2.3%.

in the circuitry, along with charge feed-through of any switches used in the configuration. Previous experiments have shown an accuracy as low as 1pA and as high as 10nA at a base current of 1uA.

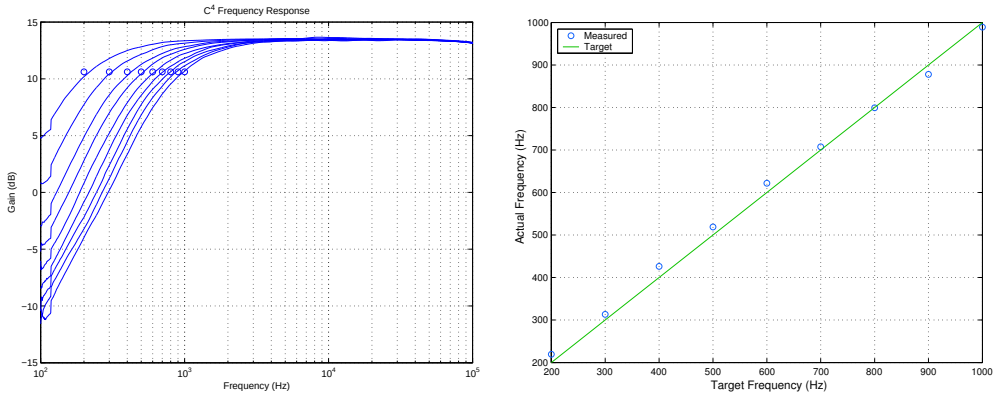
In addition to measurement errors, the device parameters are also dependent upon the operating condition of the device. For example, κ is known to vary between the sub-threshold and above-threshold regions. This variance on kappa is between 1-2%, which also introduces limitations to our predictive algorithms. One solution to the variation in parameters given operating range, would be to use an algorithm that continuously updated the parameters given a desired change, vs. the actual change. The parameter differences could then be extrapolated back out of the original equation to account for the differences. This type of approach would have the advantage of being self-updating/optimizing. This approach would require additional computational power, and thus currently does not lend itself very well to an on-chip implementation.

Figure 22 was done assuming κ doesn't change in sub-threshold region. With this assumption, we were able to achieve an accuracy of greater than 90%. Many of our test applications require an error of 95% or higher. Taking into account the change in κ we are able to achieve accuracies of greater than 98% as shown in Figure 20(b).



(a) (b)

Figure 21. Band pass filter corner frequency characterization. a.) Characterization plot of C^4 Low corner frequency for decreasing values of $iTau_{Low}$. Showing programming across the range of frequencies. b.) Characterization plot of C^4 corner frequency vs. bias current for the low corner.



(a) (b)

Figure 22. a.) C^4 plot of Low corner frequency for decreasing value of $iTau_{Low}$. All points are generated using the Kappa Projection Algorithm (KPA). b.) Error plot of the projected frequency points. Maximum error = 8%.

Due to measurement limitations, we were not able to measure the currents at the extremely low currents. Instead, we extrapolated the current given a linear relationship between current and the corner frequency of an analog filter called the C^4 [78].

The characterization plot in Figure 21(b) shows there is a linear relationship between bias current and corner frequency. Assuming this curve continues to be linear, and extrapolating the effective current from the measured frequency response, we were able to easily

Table 2. Summary of Kappa Projection Algorithm (KPA) performance

Technology	0.5μ N-well CMOS
Array size	10×32
Programming mechanisms	Hot electron injection and electron tunneling
Programming error	< 2.0% at 10pA < 10.0% at 100fA

measure currents in the 200fA range with limiting factors being the noise floor for the currents in the device which should be on the order of the reverse bias junction currents, 1 fA or less.

2.5 Floating-gate Memory Arrays

The floating-gate core used in the computational arrays differ from those used in other analog memory circuits such as Epots [19]. Epots are made ”User-Friendly” by the addition of several control circuits around each floating-gate element such that the overall circuit block is one order of magnitude larger than our floating-gate elements. Analog computational memories are a viable method of storing analog values and due to their relatively small size, lend themselves well to being scaled up as sizeable analog memory arrays. The benefit of large floating-gate computational arrays is the compactness of each core element and therefore support circuits are moved to the array periphery. Large arrays of these devices can be easily integrated with standard CMOS analog design, making them ideal for use as bias devices for filters, multipliers and many other applications.

Isolation within an array is critical to controlling the thousands, even millions, of programmable elements that may be on a chip. To that end, a functional ”AND” is done for each element. Injection requires two conditions to be satisfied within the device for injection to occur, namely drain current and sufficient drain-to-source voltage. The drain current ensures that there are carriers in the channel. The drain-to-source voltage ensures that there

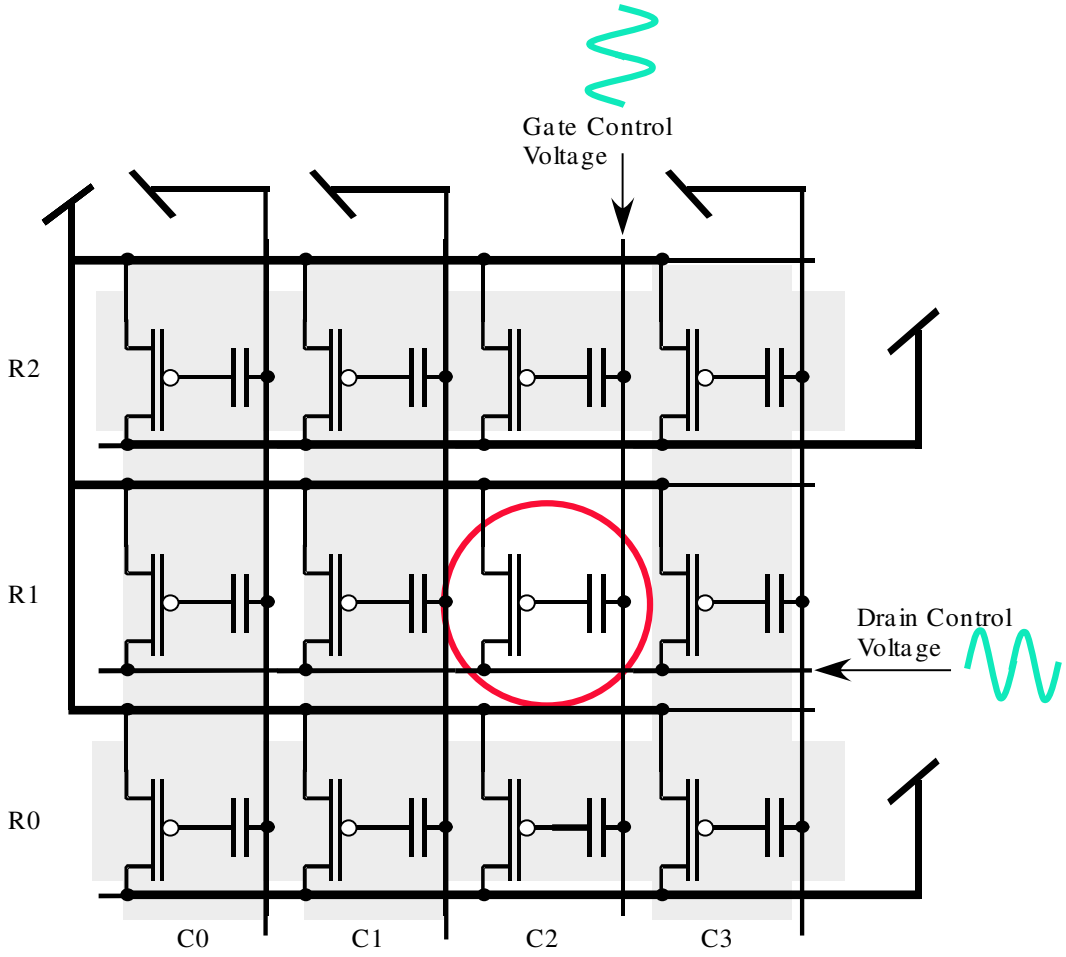


Figure 23. Floating-gate array demonstrating element isolation by controlling the gate and drain voltage of each column and row respectively. Selection of gate and drain voltages are controlled by on-chip switching circuitry. The gate, source and drain voltages are controlled by DACs on a custom programming board.

is a sufficient electric field from the drain to channel region to accelerate the carriers sufficiently to cause impact-ionization. Thus, by simply using the gate voltage in the column direction and the drain voltage in the row direction, we can effectively isolate a device by ensuring that only a single device has conditions sufficient to cause injection. This is illustrated further in Figure 23.

To ensure isolation, within an array each device is connected through various control circuits to switch a single element into and out of program mode where it's gate and drain voltages can be explicitly controlled. For the aforementioned isolation reasons, tunneling

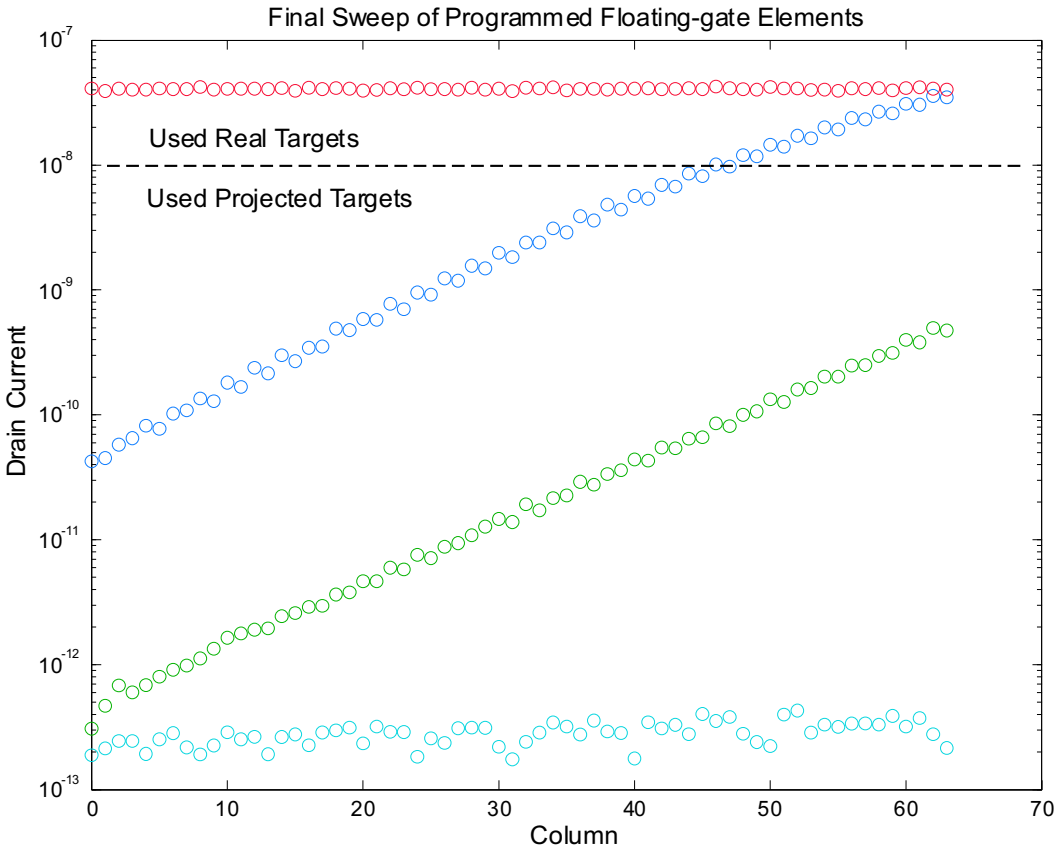


Figure 24. Early injection results showing an array of floating-gate devices programmed to exponentially spaced currents.

is used to erase the entire array and injection is used to program each element.

The isolation circuitry is made of muxes that switch the drain and gate voltages of the desired element onto a common bus for each signal. All other elements are switched to a separate voltage which ensures that those devices will not inject. The external voltages are routed off-chip and controlled by an external programming board[39]. A typical programming scenario is shown in Figure 23.

The external control circuits used to access each element and program the array are contained on a custom programming board which interfaces to a computer via a serial connection. Current versions of the programming board use FPGAs to provide the digital interface and control loop. Using these boards we have been able to program over 120dB dynamic range to an accuracy of 99%.

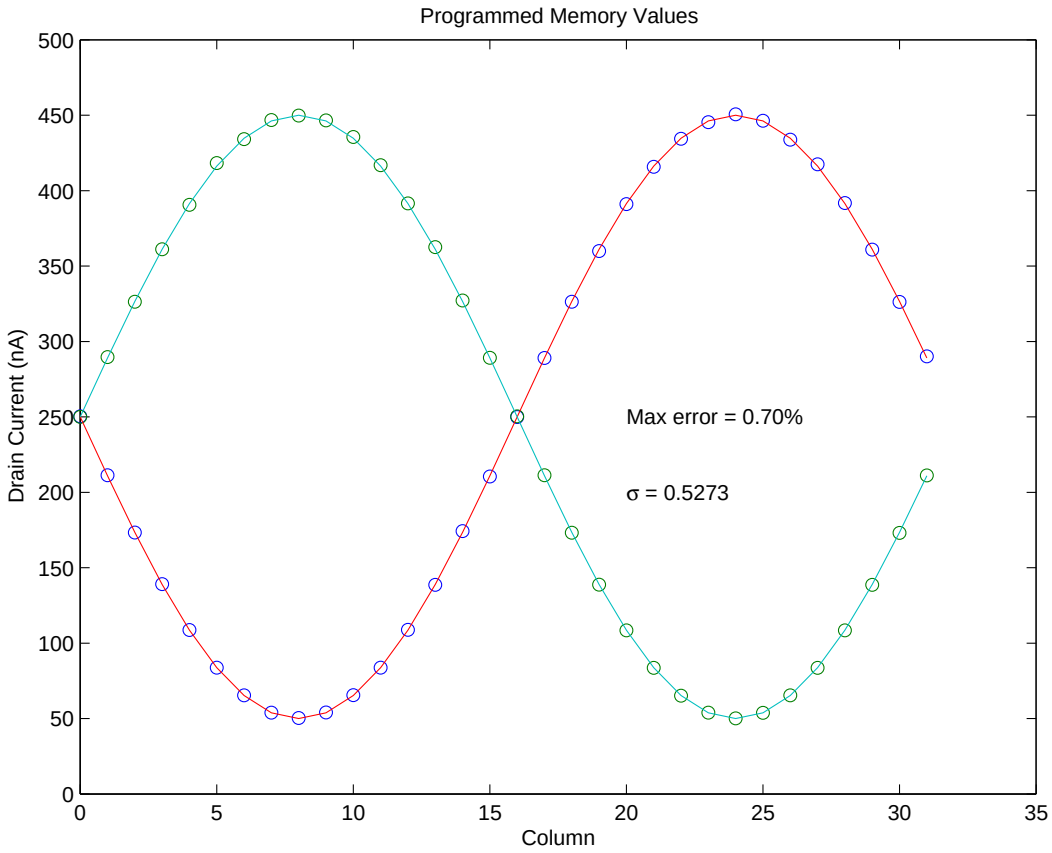


Figure 25. Injection results showing a single row of floating-gate multiplier blocks programmed to cosine coefficients. These blocks are essential to performing analog frequency transform functions. Because the values are arbitrary, one can also set these linearly or to increase or decrease logarithmically.

This chapter illustrated the floating-gate core used in the computational arrays and a programming method to accurately control the operating condition of the device. Using this element as programmable memory elements in standard CMOS analog circuits, we are able to build larger analog signal processing systems. The benefit of large floating-gate computational arrays due to ease of integration with standard CMOS analog design, making them ideal for use as bias devices for filters, multipliers and many other applications was also discussed. The next chapter will outline one of the core computational blocks used for the analog signal processing operations specifically filtering.

CHAPTER 3

PROGRAMMABLE CONTINUOUS-TIME FILTER BANKS

This chapter explores a five-transistor continuous-time band-pass filter element called the Capacitively Coupled Current Conveyer (C^4) with a programmable pass-band. We show measurement data from 1Hz to 100kHz . This chapter discusses the effects of various design parameters on frequency-range, gain and linearity. Experimental data is presented from circuits fabricated on a 0.5μm nwell CMOS process available through MOSIS.

Band-pass filter elements have a long history, from simple linear-systems to standard implementations [68]. The main use being some form of signal decomposition, whether it be to amplify/attenuate a specific signal frequency, or to separate multiple frequencies [71, 67, 30]. Tuning of these devices is critical [36, 64], in addition to matching, power, and overall die area. The filter discussed here has a simple topology (5 transistors), uses a single power-supply, uses very little power, and is easily tunable. It will also serve as a good starting point for developing higher order bandpass filters.

3.1 A Compact Band-pass Filter Element

The filter used in the programmable filter array is based on the capacitively coupled current conveyer (C^4) that has been presented [41] and characterized [78] elsewhere. Two C^4s are shown in Figure 50 on page 73 separated by a unity-gain buffer. The entire circuit of Figure 50 is called the C^4 second-order section (C^4SOS) and is the basic filter of this programmable filter bank. Since the C^4SOS is simply a cascade of two high-gain C^4s, understanding the C^4 allows analysis of the C^4SOS to be straightforward.

The initial theory on the capacitively coupled current conveyer (C^4) was developed from the Autozeroing Floating-Gate Amplifier (AFGA) [23], which had widely separated corner frequencies due to limitations of the device. The C^4 is a capacitively based bandpass filter that models the tunneling and the hot-electron injection with transistors, thus removing the

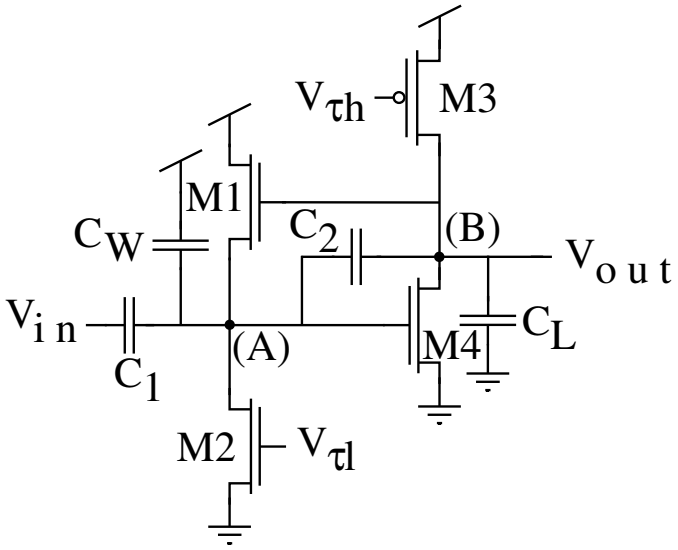


Figure 26. Schematic of a single C^4 structure. The capacitors model all explicit and parasitic capacitances in the signal path of the circuit.

corner frequency limitations. The circuit is shown in Figure 26. This circuit has previously been used in many systems [43, 30], but these applications were designed with very low Q's, typically less than 1. Such low values of Q will not work for some applications, one example of this would be a cochlea stage which requires Q's as high as 30 for low amplitude signals. In designing systems with moderate to high Q's there are certain properties that will play a significant role. Within this chapter, we hope to cover these issues to clarify the design and use of the C^4 within these systems.

Figure 26 illustrates our bandpass filter element, which serves as the initial building block for our spectral decomposition operations. The size of the block is critical since there will be one element required for each sub-band decomposition. Typical applications have used 32 sub-bands with an overall size of 800um X 100um making them easy to integrate into larger analog signal processing systems.

Here we develop the transfer function of the C^4 filter using large-signal analysis. From

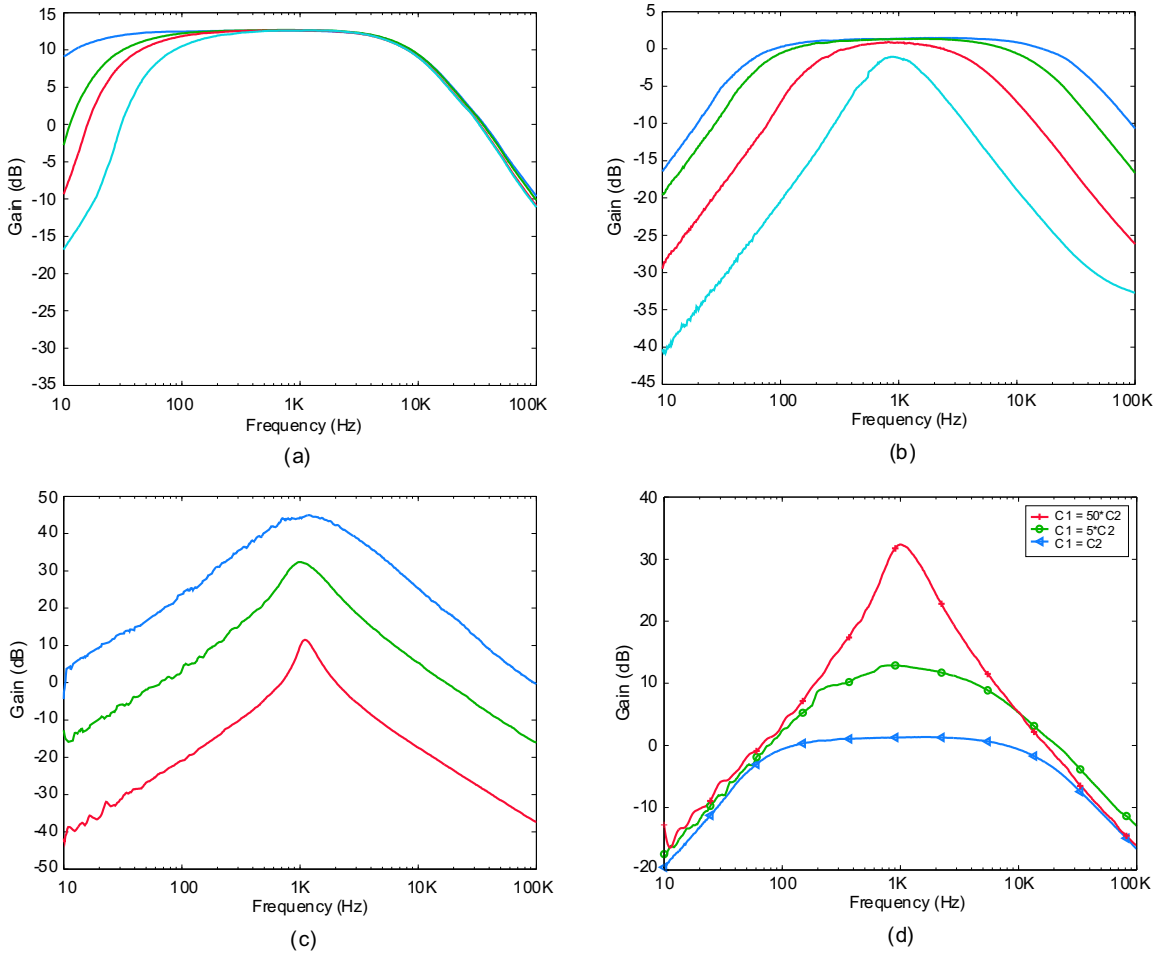


Figure 27. C^4 frequency response curves. (a) Frequency response curve for a C^4 with a gain of 5. Showing that each corner frequency can move independently. (b) Unity gain C^4 showing tunability over a wide range of frequencies. (c) The Q for a C^4 can also be tuned, up to the maximum theoretical Q. (d) C^4 change in gain with C2.

Figure 49, the differential equations for the two nodes (A) and (B) respectively are

$$C_1\frac{d(V_{in}-V_{fg})}{dt}+C_W\frac{d(-V_{fg})}{dt}+C_2\frac{d(V_{out}-V_{fg})}{dt}=I_{M2}[e^{(\kappa\Delta V_{out}-\Delta V_{fg})/U_T}-1]$$
$$C_2\frac{d(V_{fg}-V_{out})}{dt}+C_L\frac{d(-V_{out})}{dt}=I_{M3}[1-e^{-\kappa\Delta V_{fg}/U_T}e^{-\Delta V_{out}/V_A}] \quad (7)$$

If M4 is a long-channel device ($> 5\lambda$), we assume a large early Voltage (> 25) for that device and that the change in V_{out} will be small compared to this. This allows us to make the following assumption

$$e^{\Delta V_{out}/V_A}\approx 1$$

where V_A is the early voltage of M4. This assumption holds to within 4% when ΔV_{out} is

close to it's largest swing of $1000mV$, but is much more reasonable at 0.4% with voltage swings closer to more realistic values such as $100mV$. But we must keep in mind that even this assumption has minimized our SNR to -47.9dB due to this linearization technique.

Reorganizing the terms, the equations become

$$C_1\frac{dV_{in}}{dt} + C_2\frac{dV_{out}}{dt} - (C_1 + C_2 + C_W)\frac{dV_{fg}}{dt} = I_{M2}[e^{-(\kappa\Delta V_{out} - \Delta V_{fg})/U_T} - 1] \tag{8}$$

$$C_2\frac{d(V_{fg})}{dt} - (C_2 + C_L)\frac{dV_{out}}{dt} = I_{M3}[1 - e^{-\kappa\Delta V_{fg}/U_T}] \tag{9}$$

At this point we must make further assumptions to even begin to solve this system of equations. We will begin by making two undesirable, yet necessary assumptions at this point. The errors introduced by linearizing at this stage will be discussed later when looking at distortion. First we assume

$$-(\kappa\Delta V_{out} - \Delta V_{fg})/U_T \ll 1 \tag{10}$$

from 8. However, we have already seen that ΔV_{out} can be as large as $1000mV$. Also, given that $U_T \sim 25mV$ this makes this assumption even less accurate. Next we assume that

$$-\kappa\Delta V_{fg}/U_T \ll 1 \tag{11}$$

from equation 9. Keep in mind that ideally, ΔV_{fg} would be zero if the follower in the feedback path is indeed operating correctly, which means this approximation holds during normal operating conditions. For simplicity we will continue to use both of these assumptions and make the following simplification

$$e^x - 1 \approx x$$

$$1 - e^{-x} \approx x$$

Transforming (8) and (9) into

$$C_1\frac{dV_{in}}{dt} + C_2\frac{dV_{out}}{dt} - (C_1 + C_2 + C_W)\frac{dV_{fg}}{dt} = -I_{M2}\frac{\kappa\Delta V_{out} - \Delta V_{fg}}{U_T} \tag{12}$$

$$C_2\frac{d(V_{fg})}{dt} - (C_2 + C_L)\frac{dV_{out}}{dt} = I_{M3}\frac{\kappa\Delta V_{fg}}{U_T} \tag{13}$$

By taking the Laplace transforms of (12) and (13), these equations become

$$sC_1V_{in} + sC_2V_{out} - s(C_1 + C_2 + C_W)V_{fg} = -\frac{\kappa I_{M2}}{U_T}V_{out} + \frac{I_{M2}}{U_T}V_{fg} \tag{14}$$

$$sC_2V_{fg} - s(C_2 + C_L)V_{out} = \frac{\kappa I_{M3}}{U_T}V_{fg} \tag{15}$$

The sum of capacitances is renamed as

$$C_T = C_1 + C_2 + C_WC_O = C_2 + C_L$$

for the total capacitance and the output capacitance, respectively. Again, rewriting the equations, they become

$$sC_1V_{in} + sC_2V_{out} - sC_TV_{fg} = -\frac{\kappa I_{M2}}{U_T}V_{out} + \frac{I_{M2}}{U_T}V_{fg} \tag{16}$$

$$sC_2V_{fg} - sC_OV_{out} = \frac{\kappa I_{M3}}{U_T}V_{fg} \tag{17}$$

By defining two time constants as

$$\tau_l = \frac{C_2U_T}{\kappa I_{\tau_l}}, where I_{\tau_l} = I_{M2}$$

$$\tau_f = \frac{C_2U_T}{\kappa I_{\tau_h}}, where I_{\tau_h} = I_{M3}$$

(16) and (17) become

$$s\frac{C_1}{C_2}\tau_lV_{in} + s\tau_lV_{out} - s\frac{C_T}{C_2}\tau_lV_{fg} = -V_{out} + \frac{1}{\kappa}V_{fg}$$

$$s\tau_fV_{fg} - s\frac{C_O}{C_2}\tau_fV_{out} = V_{fg} \tag{18}$$

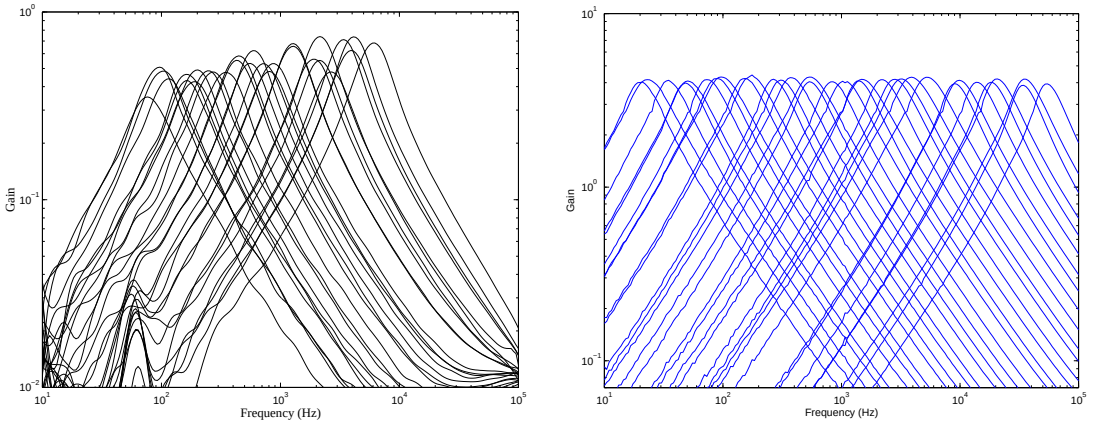
Rearranging these two expressions,

$$V_{in}\left(s\frac{C_1}{C_2}\tau_l\right) + V_{out}(s\tau_l + 1) = V_{fg}\left(s\frac{C_T}{C_2}\tau_l + \frac{1}{\kappa}\right)$$

$$V_{fg}\left(s\tau_f - 1\right) = s\frac{C_O}{C_2}\tau_fV_{out} \tag{19}$$

The C^4 is a capacitively based bandpass filter with electronically tunable corner frequencies that can be set independently of one another. The frequency response of the C^4 is governed by

$$\frac{V_{out}}{V_{in}} = -\frac{C_1}{C_2}\frac{s\tau_l(1 - s\tau_f)}{s^2\tau_h\tau_l + s(\tau_l + \tau_f(\frac{C_o}{\kappa C_2} - 1)) + 1} \tag{20}$$



(a) (b)

Figure 28. C^4 biasing using resistors and programmable devices, respectively. a.) Array of 32 C^4 filters biased using a resistor ladder. Transistor mismatch is significant and the filter taps are not even monotonic. b.) Array of 32 programmable C^{4}s biased using programmable memories with exponentially spaced currents within 95% accuracy. By adding programming to the filter bank, qualitatively one is able to quickly see a significant improvement.

where the time constants are given by

$$\tau_l = \frac{C_2 U_T}{\kappa I_{\tau_l}} \qquad \tau_f = \frac{C_2 U_T}{\kappa I_{\tau_h}} \qquad \tau_h = \frac{C_T C_O - C_2^2}{C_2} \frac{U_T}{\kappa I_{\tau_h}}$$

and the total capacitance, C_T, and the output capacitance, C_O, are defined as $C_T = C_1 + C_2 + C_W$ and $C_O = C_2 + C_L$. The currents I_{τ_l} and I_{τ_h} are the currents through M_2 and M_3, respectively in Figure 26. With normal usage, τ_f is so fast that the zero it produces lies far outside of the operating range. Hence, the C^4 takes on the form of a bandpass filter within the region of interest with ±20 dB/decade slopes outside the passband. The midband gain is $-C_1/C_2$. U_T is the thermal voltage ($25.9mV@25^oC$) and κ is the subthreshold slope. The plots of Figure 27 show measured data from a $0.5\mu m$ process available through MOSIS that summarizes the frequency response of the C^4.

The C^4 takes on the properties of a bandpass filter with first-order roll-off and a passband gain set by the ratio of the two coupling capacitors as $A_v = -C_1/C_2$. The overlap capacitance of the MOSFET causes there to always be some effective C_2 capacitance, even if it is only a few fF, so the gain is not infinite. The overall time constant of the filter, which

gives the center frequency, is

$$\tau = \sqrt{\tau_l \tau_h} \tag{21}$$

Furthermore, since the corner frequencies of the C^4 are completely independent of each other, as shown in Figure 27a, either one of the corner frequencies could be pulled to the extreme allowing the C^4 to take the form of a lowpass or highpass filter.

Figure 28 (a) shows a C^4 array biased using a resistor ladder. Transistor mismatch is significant and the filter taps are not even monotonic. Figure 28 (b) shows an array of 32 programmable vanilla C^4s programmed with exponentially spaced corner frequencies within 95% accuracy. By adding programming to the filter bank, qualitatively one is able to quickly see an improvement. Further improvements require an additional calibration step.

3.2 Designing for Q

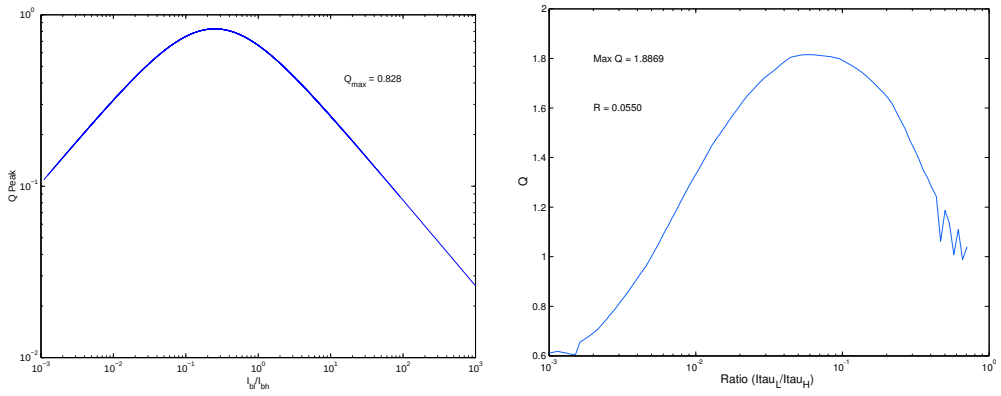
By tuning the filter such that $\tau_h > \tau_l$, resonance occurs, and the value of the Q peak is

$$Q = \sqrt{\frac{C_T C_O - C_2^2}{C_2^2}} \frac{x}{1 + x^2\left(\frac{C_O}{\kappa C_2} - 1\right)} \tag{22}$$

where

$$x = \sqrt{\frac{I_{\tau l}}{I_{\tau h}}} \tag{23}$$

Figure 29 is a plot of the Q peak versus the ratio of I_{τ_l}/I_{τ_h} for a unity-gain C^4 using the extracted values of the capacitances from the layout. As can be seen from this MATLAB plot, there is a maximum value that the resonance can achieve. The maximum value occurs when I_{τ_h} is slightly larger than I_{τ_l} for the capacitances of the fabricated circuit that yielded the plots of Figure 29. The maximum value of the Q peak can be predicted for a certain set of capacitances by taking the derivative of (22) and finding the maximum, and this is given



(a) (b)

Figure 29. Simulation and measurement data of filter "Q" vs. current ratio. a.) Calculated Q as a function of I_{τ_h}/I_{τ_l}. There is a maximum Q at a particular current ratio, and the curve is relatively flat around that maximum point. The maximum achievable Q for this circuit was designed to be approximately 1. Higher maximum values of Q can be explicitly designed into the circuit. b.)Experimental data of maximum achievable "Q" for a given filter.

by

$$Q_{max} = \frac{1}{2}\sqrt{\frac{\kappa\left(C_T C_O - C_2^2\right)}{C_2\left(C_O - \kappa C_2\right)}} \tag{24}$$

$$\approx \frac{1}{2}\sqrt{\frac{\kappa\left(C_1 + C_W\right)}{C_2}} \quad \text{for} \quad C_2 \ll C_O \tag{25}$$

This represents the maximum Q for small amplitudes and will drop as amplitude increases. This phenomena is not discussed within the scope of this paper, but will be discussed later. As can be seen from Figure 29, the C^4 was not designed to have a large Q peak. However, by changing the capacitances, a much larger Q peak can be achieved. For example, by reducing the value of C_2, more resonance occurs. A good method to increase Q is to not explicitly draw any C_2 capacitance and use the gate-drain overlap capacitance of M4 in Figure 26 as C2.

Removing the feedback capacitor C_2 transforms the C^4 into a high-gain filter since the effective value of C_2 is simply the overlap capacitance of M_4. This is a configuration is called a vanilla C^4. In addition to being high gain, a vanilla C^4 has increased resonance that can be expressed by

$$Q_{max} \approx \frac{1}{2}\sqrt{\frac{\kappa\left(C_1 + C_W\right)}{C_2}}. \tag{26}$$

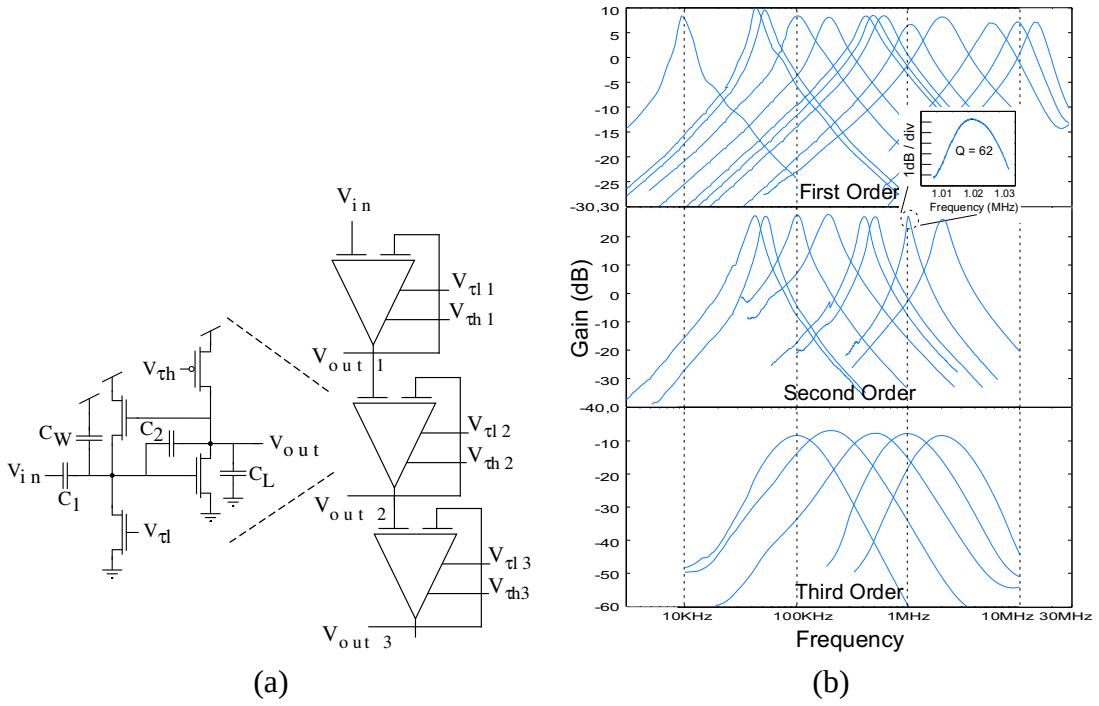


Figure 30. (a) C^4 circuit and a cascade structure to generate higher order filters. The center frequency, along with the Q for each filter stage is programmable by integrating programmable elements into the circuit structure. (b) The frequency response for a various filter structures designed for a 1st-order up to a 3rd-order filter response. The 2nd order stage was designed to have a very high Q and a Q of 62 was measured, which is ample for most auditory applications.

The larger the *Q* peak, the greater the isolation of the center frequency when the vanilla C^4 is tuned to have a narrow bandwidth.

3.3 Designing for multiple filter stages

These filters can be cascaded to increase the stop-band rolloff, as well as the overall Q of the filter. Operation of the C^4 Second-Order Filter is similar to a vanilla C^4 since the C^4 Second-Order Filter is simply a cascade of two vanilla C^4s isolated by a buffer. By tuning each of the C^4s comprising the C^4SOS to have identical time constants, the overall response of the C^4SOS has ±40 dB/decade slopes outside the passband and a potentially large *Q* peak.

One potentially hazardous trait of the C^4 is that the input capacitance of the circuit does not necessarily remain constant, but it varies based on frequency. This is particularly

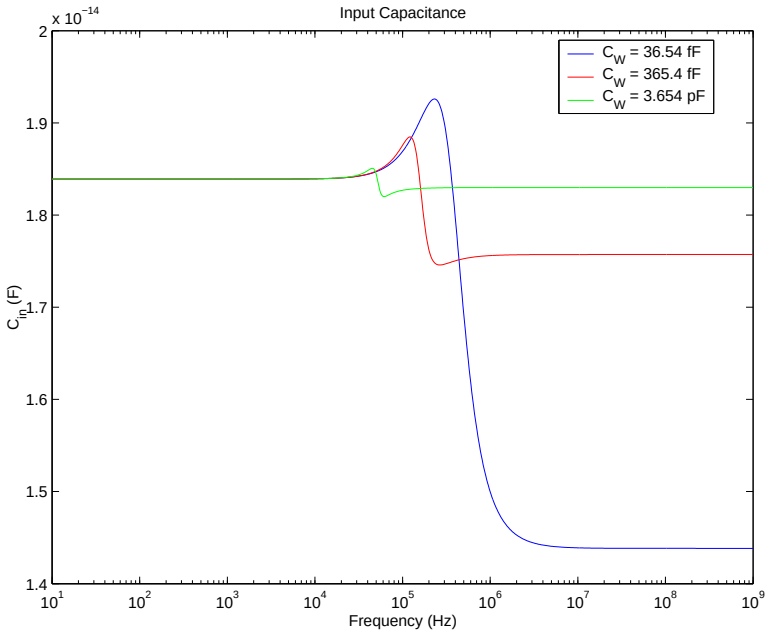


Figure 31. Change in C^4 input capacitance vs. frequency. Low frequency input capacitance = C_1. This transitions to the high-frequency input capacitance which is roughly $C_1//C_w$.

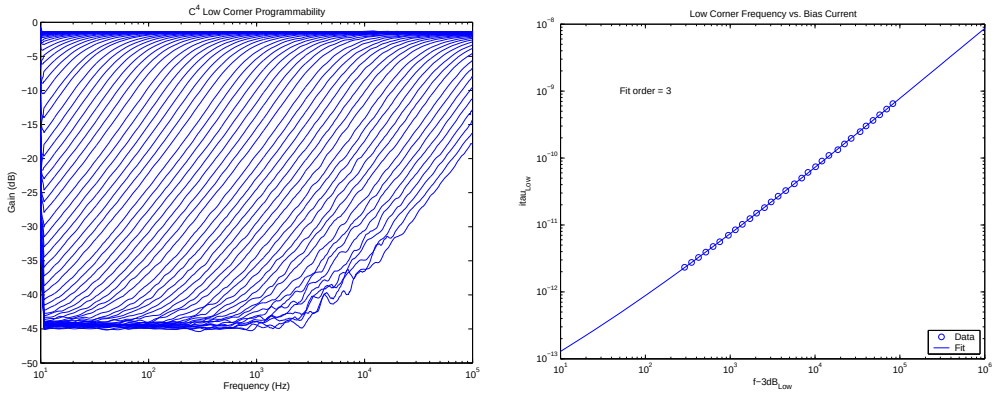
critical when cascading multiple C^4 stages to create higher order filter banks. If the previous stage in the overall system is dependant upon the input capacitance of the C^4 for its load capacitance, then this could be a serious issue.

Referring back to Figure 26, the input capacitance can easily be found for the cases of very low frequencies and very high frequencies. For very low frequencies, the middle node is an AC ground because of the high-gain amplifier. Hence, the input capacitance for low frequencies is simply $C_{in} = C_1$. For very high frequencies, the transistors can no longer follow the signals, so the C^4 reduces to a network of capacitors and the input capacitance becomes the series/parallel combination of the capacitances in this network seen looking into the input, V_{in}. The input capacitance for the two extreme cases are given by

$$C_{in}(f \rightarrow 0) = C_1$$

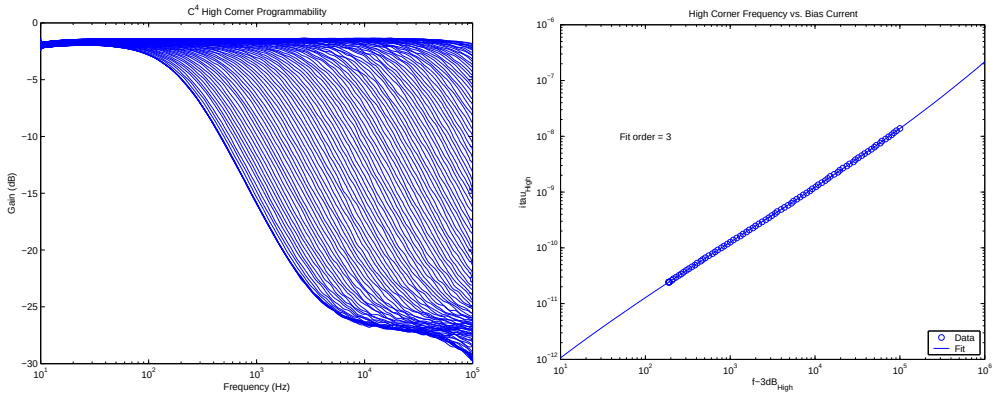
$$C_{in}(f \rightarrow \text{inf}) = C_1 || (C_W + C_2 || C_L) \approx C_1 || C_W \tag{27}$$

This approximation holds in the case when C_2 is significantly smaller than the load capacitance, C_L. While the input capacitance is well defined for very low frequencies and for



(a) (b)

Figure 32. a.) Characterization plot of C^4 Low corner frequency for decreasing values of $iTau_{Low}$. Showing programming across the range of frequencies. b.) Characterization plot of C^4 corner frequency vs. bias current for the low corner.



(a) (b)

Figure 33. a.) Characterization plot of C^4 High corner frequency for increasing value of $iTau_{High}$. Showing programming across the range of frequencies. b.) Characterization plot of C^4 corner frequency vs. bias current.

very high frequencies, and since these values are not identical, there is a transition region for the input capacitance between these states and this figure shows that the transition region between these two values of C_{in} which occurs over a confined frequency band near the center frequency of the C^4. Figure 31 shows results of a SPICE simulation in which the input capacitance was computed. Simulation and experiments closely agree for this circuit.

However, since the width of the swing in input capacitance is a function of the values of the capacitances, the effect of the input-capacitance shift can be minimized with a judicious

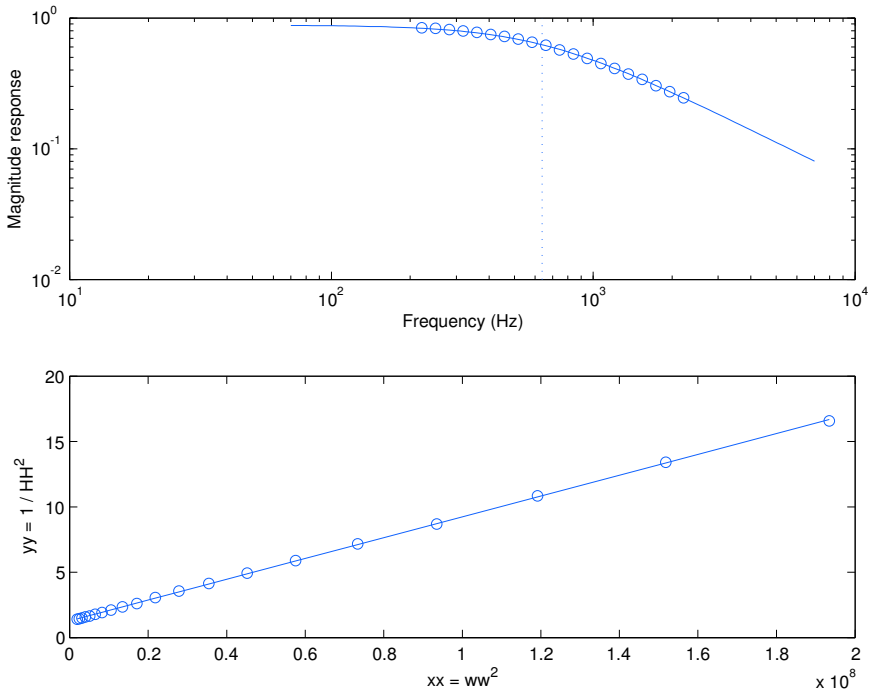


Figure 34. Extraction of C^4 frequency response parameters. A linear regression is applied to the result of the polynomialization (shown below). The solid line in the upper plot is the ideal lowpass response calculated with the parameters that have been extracted. Frequency response information is only needed in the region surrounding the corner frequency; additionally, parameters can be obtained from less dense data with nearly the same accuracy. Measuring fewer data points allows for increased speed in the tuning algorithm.

choice of capacitor values. Increasing the drawn size of C_W is the best choice for reducing the shifting input capacitance for the simple reason that the larger the value of C_W, the more closely $C_1 \| C_W \approx C_1$, and hence the more closely the high-frequency C_{in} approaches the the low-frequency C_{in}. Figure 31 shows the results of 10-fold increases in the capacitance of C_W. The larger that C_W is drawn, the less the effect of the input capacitance shift on the system.

Another way of reducing the effects of the shifting input capacitance is to put a buffer in front of the C^4 so that the previous circuit always sees the same load capacitance. However, for many cases, spending the real estate required for a buffer could be better used by simply increasing the size of the C_W capacitor because increasing C_W has advantages besides keeping C_{in} relatively fixed. The primary reason that C_W is drawn in the C^4 is to

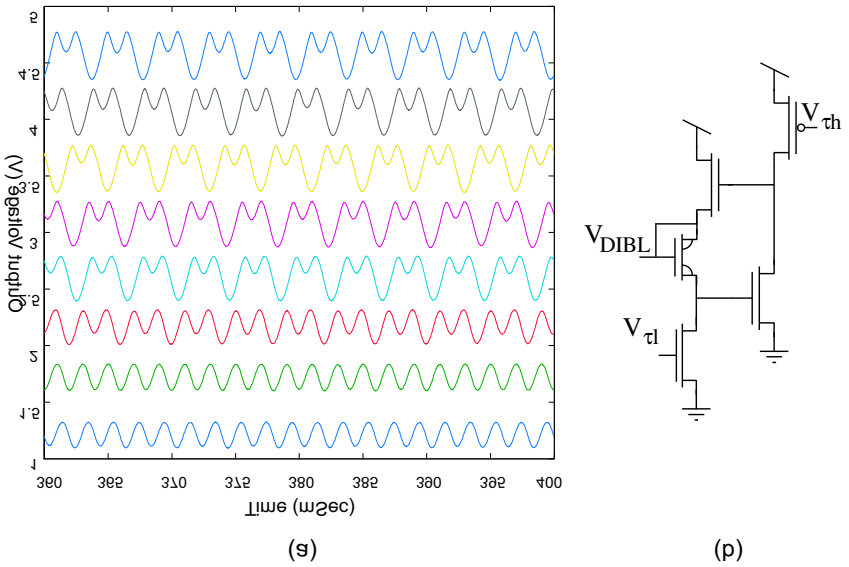


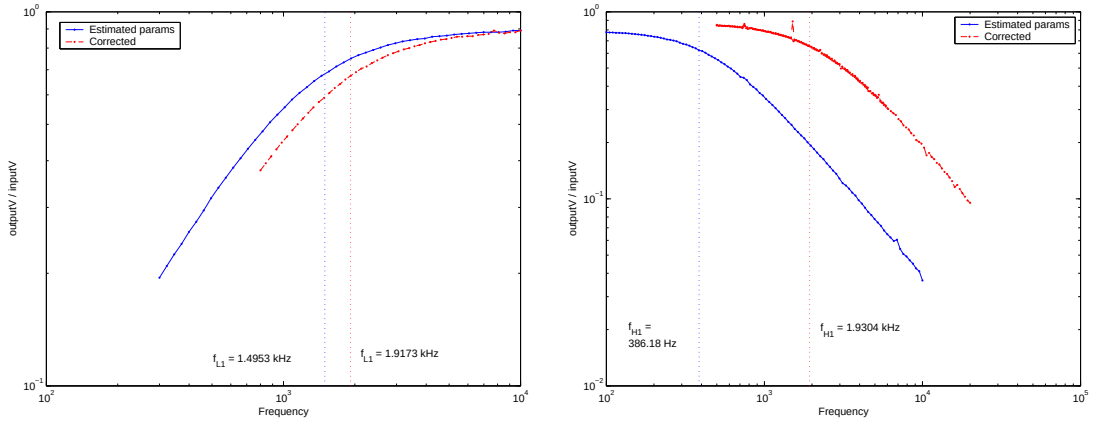
Figure 35. (a) C^4 change in linearity with DIBL voltage. The linear range of a C^4 bandpass filter is improved by using source degeneration in the feedback path. A DIBL transistor is chosen to provide an exponential relationship that is more flat than a regular transistor because of it's very small early voltage. For simplicity each curve is offset +500mV. The lower curve shows the output when the DIBL voltage is closest to it's ideal voltage. (b) A circuit implementation to provide a self-bias for the DIBL transistor. The optimal DIBL voltage would be determined by the bias point of a given C^4, therefore a separate circuit would be required for each filter.

capacitively divide the input signal and thus increase the input linear range.

3.4 Decreasing Distortion

The C^4 uses a short-channel device, named a DIBL (Drain-Induced Barrier Loading) transistor, to decrease the loop-gain in the feedback path and increase the overall linearity of the device. A DIBL transistor is chosen to provide an exponential relationship that is more flat than a regular transistor because of its very small early voltage. During initial characterization this device has its gate pulled to VDD in order to remove it from the loop, however, an optimal bias point for the device is shown to improve linearity. Figure 35(a) shows the decrease in the second harmonic as the DIBL bias approaches its optimal value. The optimal DIBL voltage is dependent on the DC operating point, which changes with corner frequency for each filter. Figure 35(b) shows a self-biasing scheme that is currently being testing to work over all frequencies thus increasing linearity without external biases.

In the case of an array of these devices, the self-biasing DIBL circuitry would be required for each filter tap because V_{DIBL} is dependent upon $V_{\tau l}$ and $V_{\tau h}$, which change depending on the frequency response of the individual filter.



(a) (b)

Figure 36. Example of applying correction factor to (a) low frequency and (b) high frequency corners. The correction factor is calculated as a ratio of the actual time constant to the desired time constant and is used in determining the correct bias current to program.

3.5 Programmed Filter Bank

An array of 32 C^4s was fabricated with a 0.5μm process available through MOSIS. While the bandpass filter elements can be programmed to any desired center frequency spacing and bandwidth, we will use the example of exponentially spaced center frequencies with narrow bandwidths and moderate Qs as this is a highly advantageous configuration in audio signal processing. This type of configuration closely models the biology of the human cochela [cite] and it lends itself to allow subbands of frequency to be independently manipulated since there is a real-time frequency decomposition occurring. Figure 28b shows the frequency response of each of the 32 filter taps. These filters were programmed so that the $I_{\tau l}$s and $I_{\tau h}$s for each filter tap had exponentially spaced currents that were programmed within 95% of their desired values.

By programming the filter bank to have exponentially spaced center frequencies with

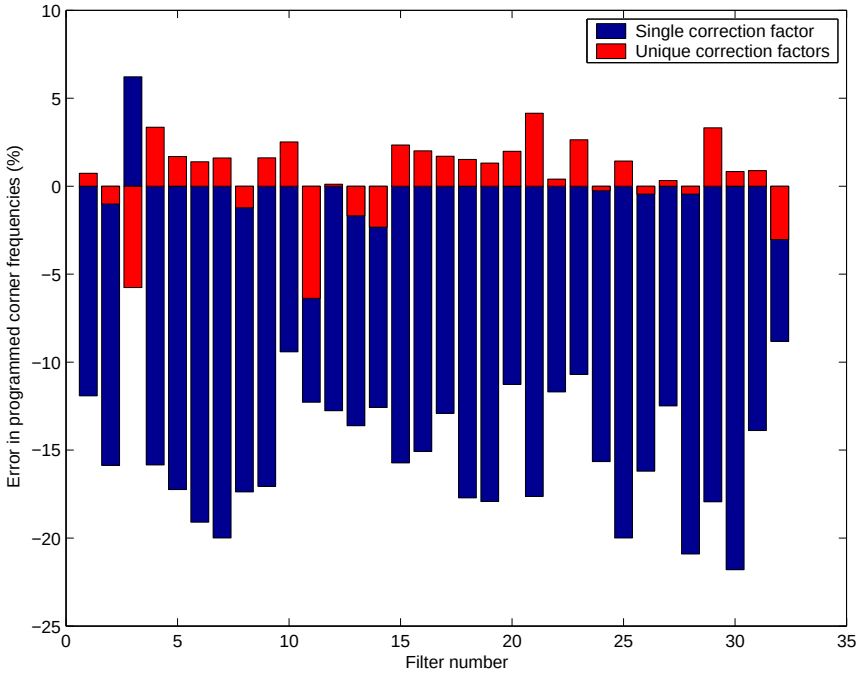
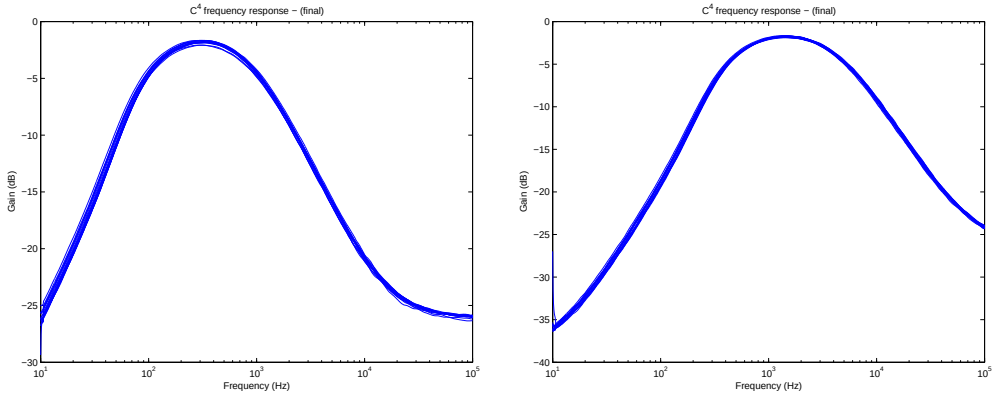


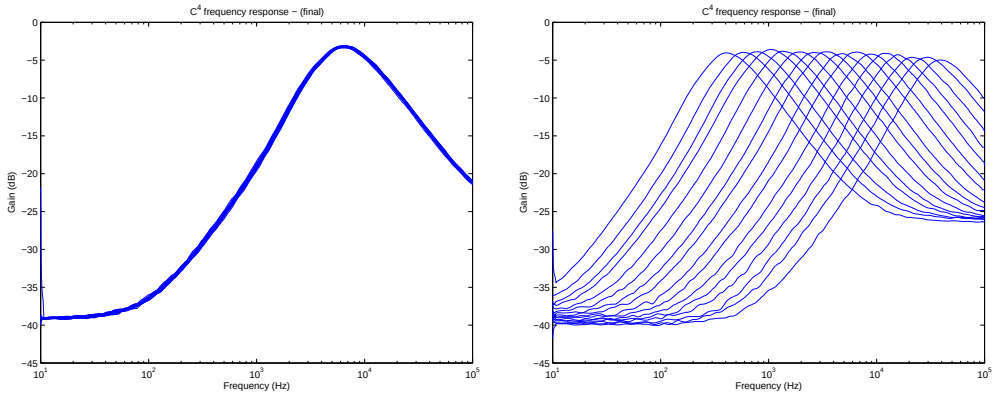
Figure 37. This plot shows the percentage error in corner frequency for a bank of filters, with a single correction factor and with multiple, adjusted correction factors. The adjusted correction factors account for both mismatch in the circuits from filter to filter, but also for offsets as a result of the programming algorithm. The final errors can be attributed to errors introduced in the programming algorithm and potential inaccuracies in the extraction of frequency response parameters.



(a) (b)

Figure 38. C^4 filter bank response data. a.) All filter corners programmed to 100 Hz and 1kHz. b.) All filter corners programmed to 500Hz and 5kHz.

narrow bandwidths and moderate resonance, an input signal is broken down into its respective frequency components. This type of filter bank is extremely useful in signal processing applications in which the specific algorithm is easier to perform on individual subbands of



(a) (b)

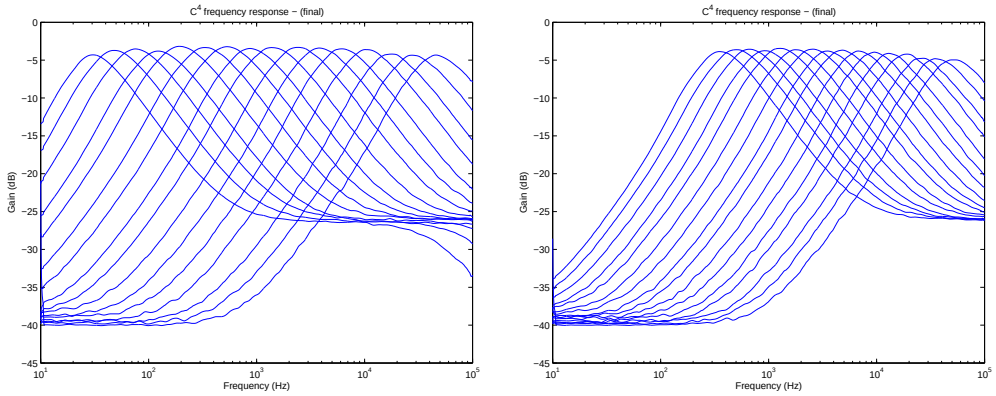
Figure 39. C^4 **filter bank response data. a.) All filter corners programmed tight to 5kHz and 10kHz. b.) Bias currents programmed to a "nice" spacing. Not an explicit frequency.**

the input signal, as is often done in auditory processing. These same signal processing techniques can also be extended into the IF band with this same filter bank. Several versions of this programmable array of bandpass filters have been fabricated using $0.5\mu m$ and $0.35\mu m$ processes available through MOSIS.

3.6 Programming Out Offsets

Figure 28 (b) illustrates one of the fundamental design issues with analog circuits, and this is that no matter how accurately biases can be set, circuit performance is affected by mismatches that occur during the fabrication process. The 32 traces from the programmable bandpass array are monotonically spaced, which is a difficult task and has only been overcome by very clever circuit design [71]. In contrast, this monotonicity and spacing was simply programmed by the floating-gate biases. However, inspection of Figure 28 (b) shows that corner frequencies are not *perfectly* spaced. However, this is of no concern because these errors due to mismatch of transitor and capacitor sizes can be simply programmed out with the floating gates.

Using the time constant equations and estimated values for the device constants and capacitances, an initial current is calculated and programmed for each bias point. At the



(a) (b)

Figure 40. First-order C^4 filter bank response data. a.) C4 filter corners programmed to have a Q=2 with the high frequency starting at 64kHz and all the other corners falling out from there. b.) C4 filter corners programmed to have a Q=3 with the high frequency starting at 64kHz and all the other corners falling out from there.

outset, the corners are chosen so as to be well-spaced to ensure that the high frequency corner is independent of the influence of the low corner's time constant, and vice versa. The magnitude of the frequency response for the filter is then measured. Frequency response parameters can be easily extracted by polynomialization of the magnitude response. This involves transforming the magnitude response data so that a linear regression can be applied and the parameters extracted. The transformation for highpass and lowpass responses (and thus low and high corners, respectively) can be written as follows:

$$y_L(x = \frac{1}{\omega^2}) = \frac{1}{|H(jw)|^2} = \frac{1}{A^2} + \frac{1}{\tau_L^2 A^2} \quad x$$

$$y_H(x = \omega^2) = \frac{1}{|H(jw)|^2} = \frac{1}{A^2} + \frac{\tau_H^2}{A^2} \quad x \tag{28}$$

where A is the passband gain. A typical frequency response for the high corner of a C^4 is shown in Figure 34 along with the regression of the resulting polynomial. Using the extracted frequency response parameters, a correction constant is calculated as a ratio of the actual time constant and the target time constant. This correction factor is then multiplied with the original target current and the floating-gate devices are reprogrammed.

This method of estimation and correction has proven very successful. Data from a

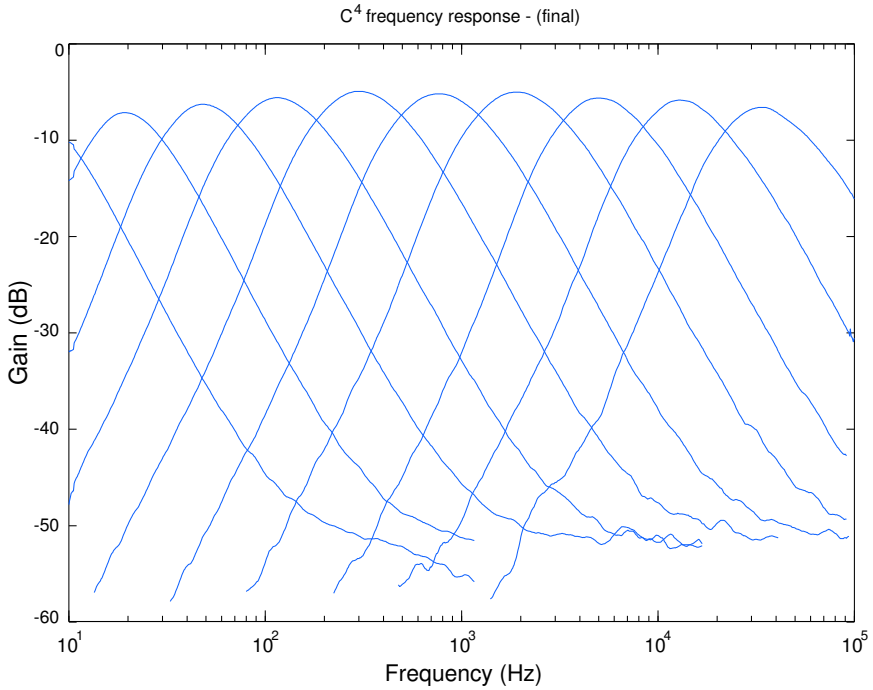


Figure 41. Second-order C^4 array programmed with a Q of 1.

programmable C^4circuit fabricated through MOSIS is shown in Figure 36. The plots show both the original and the corrected frequency response curves of the low and high frequency corners. In the case of the high frequency corner, the target corner frequency was 2.0 kHz; after one correction step, the corner frequency was measured as 1.93 kHz, a four percent error. The error was within the five percent tolerance that was characteristic of the algorithm used for floating-gate programming; it is reasonable to assume an increase in accuracy of the programming algorithm would result in a further increase in the accuracy of the tuning algorithm.

The correction factor was calculated initially for both the high and low frequency corners in only a single filter. These single correction factors were then applied to all of the filters in the filter bank with all of the corner frequencies targeted for the same value. From these results, correction factors unique to each filter were calculated and stored. The corner frequencies were then reprogrammed using the unique correction factors. Figure 37 shows the percentage error of the actual high corner frequencies versus the targets. The

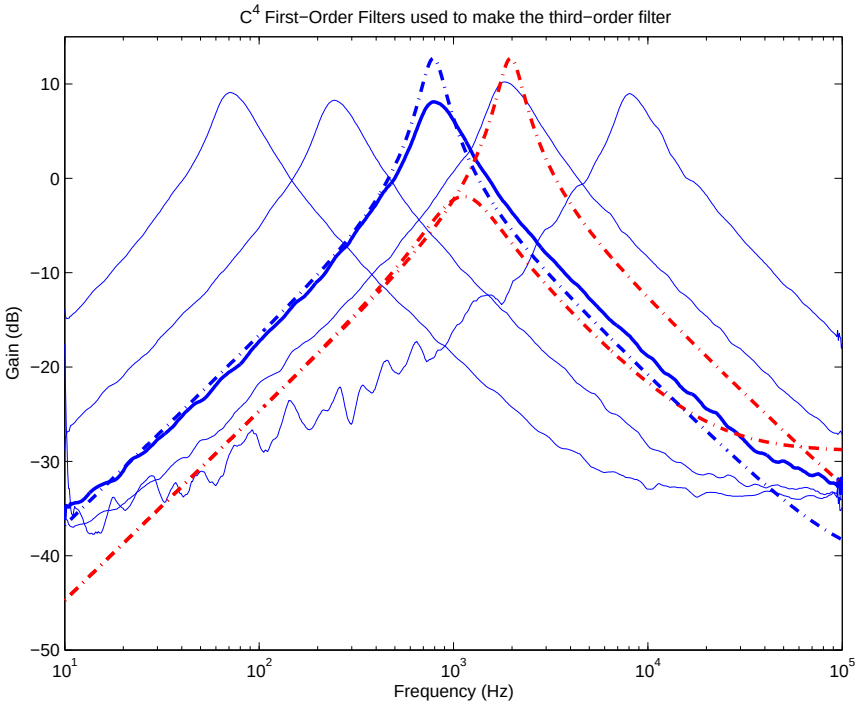


Figure 42. C^4 first-order filter bank measured and theoretical fit. The ”dark” dotted line is the theoretical filter used, while the solid line is the experimental filter response. The other two ”lighter” dotted lines are the theoretical responses used to create the third-order filter data. They do not refer to any of the other experimental curves.

plot shows data from the filter bank programmed with a single correction factor and with multiple, adjusted correction factors.

Biasing of these arrays is critical. The bias currents typically scale according to the following relationship.

$$I_{n+1} = A * I_n \tag{29}$$

where A is a scale larger than 1, and n goes from 1 to the number of sub-bands in the system. There are various ways to generate these bias values. Each with its advantages and disadvantages.

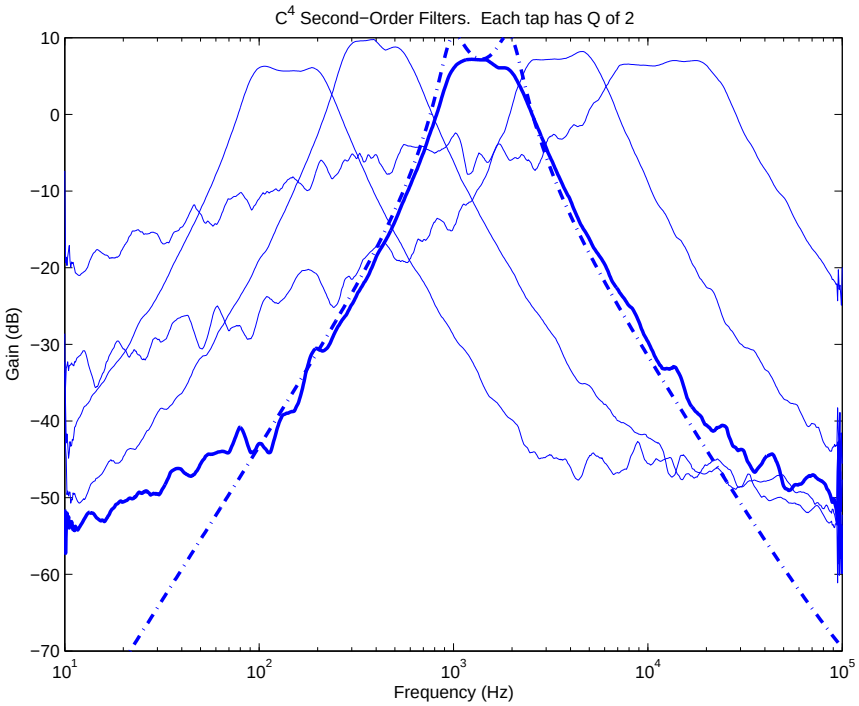


Figure 43. C^4 second-order filter bank measured data and theoretical fit. The ”dark” dotted line is the theoretical filter used, while the solid line is the experimental filter response.

3.7 Biasing Techniques

Since the bias values for each filter tap in the bank of filters is ideally related by a constant scale function, this raises some interesting questions about biasing methods for these devices. the two methods that will be discusses are biasing using the floating-gates directly and biasing using a scales biasing generator of some kind; for our purposes we have used a modified bias generator structure originally introduced in [7].

3.7.1 Floating-gate Direct Biasing

One method is using floating-gate biasing directly. This method allows the current for each filter element to be programmed independently. The benefit to this is flexibility and also space because there is less overhead associated with simply programming each device individually. Using these elements and programming has been covered in chapter 2.

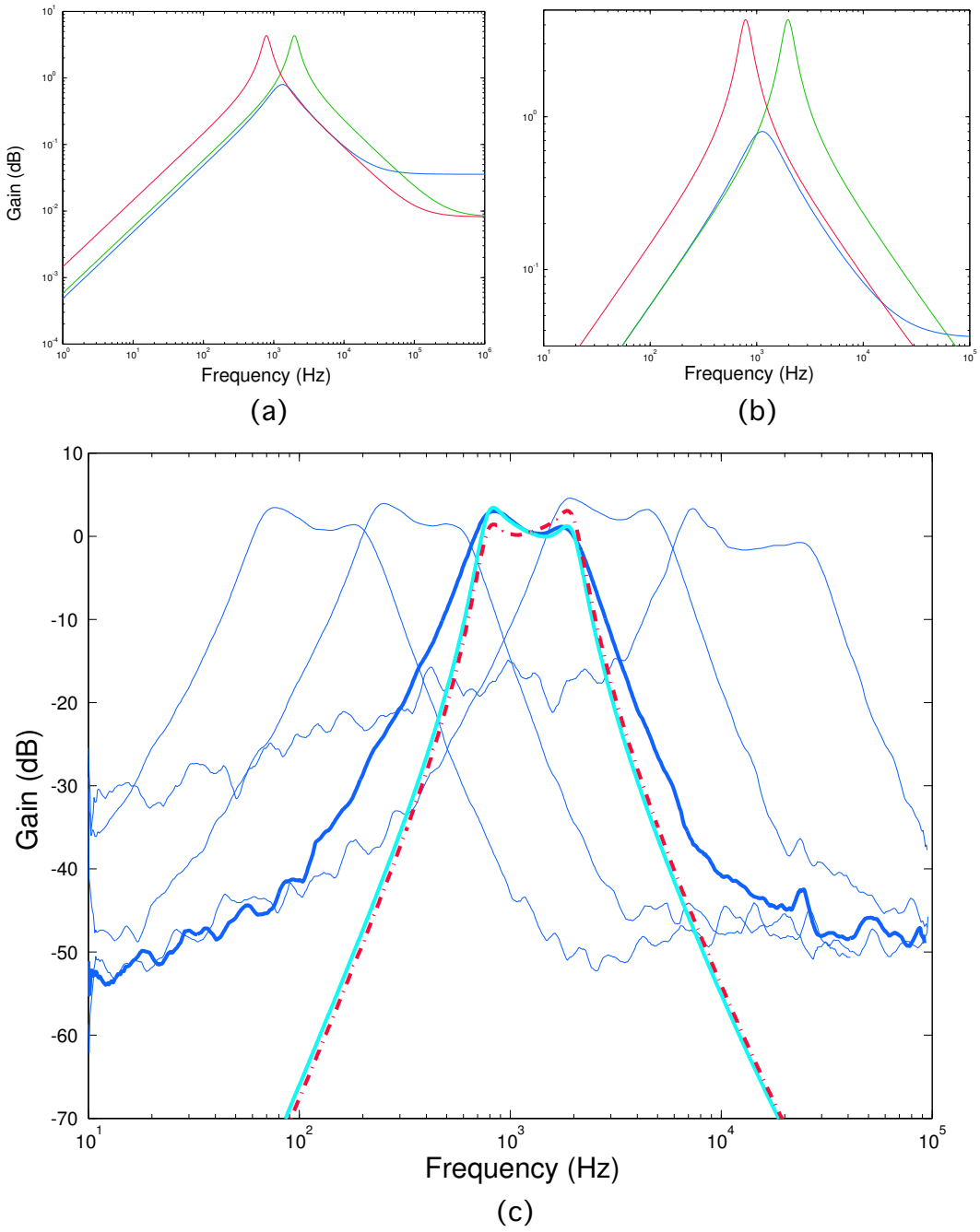


Figure 44. C^4 third-order measured data and theoretical fit. a.) Bands used to create the Third-order filter bank (data set 3). b.) Bands used to create the shifted Third-order filter bank (data set 3). c.) Third-order filter bank (data set 3) showing difference by shifting one filter.

3.7.2 Programmable Bias Generators

Because the scale is constant between all filters, one can also use Bias Generators which provide a fixed, designable scale value for an array of currents [7]. This works well for

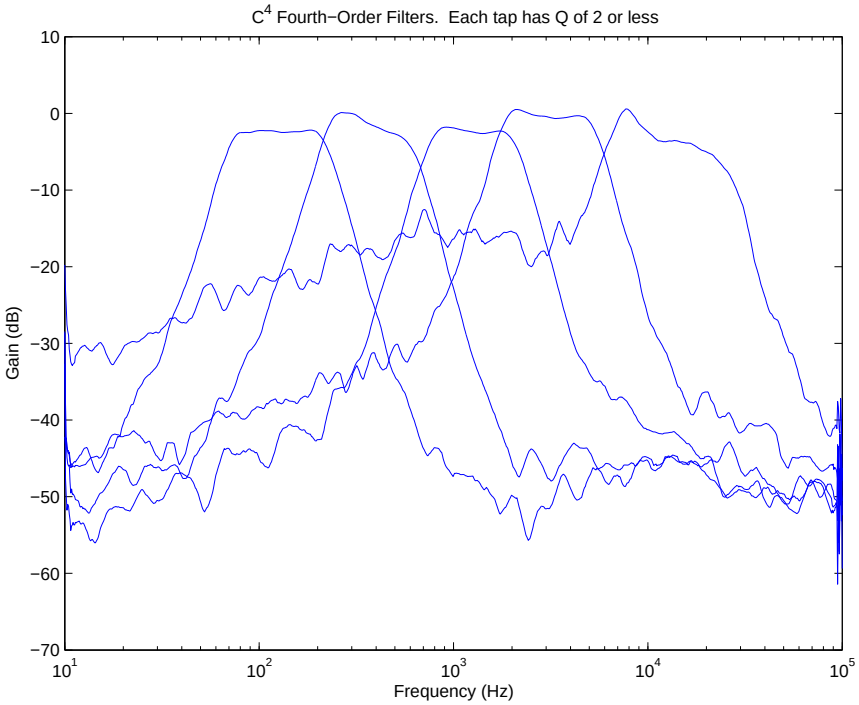


Figure 45. C^4 fourth-order filter bank response data.

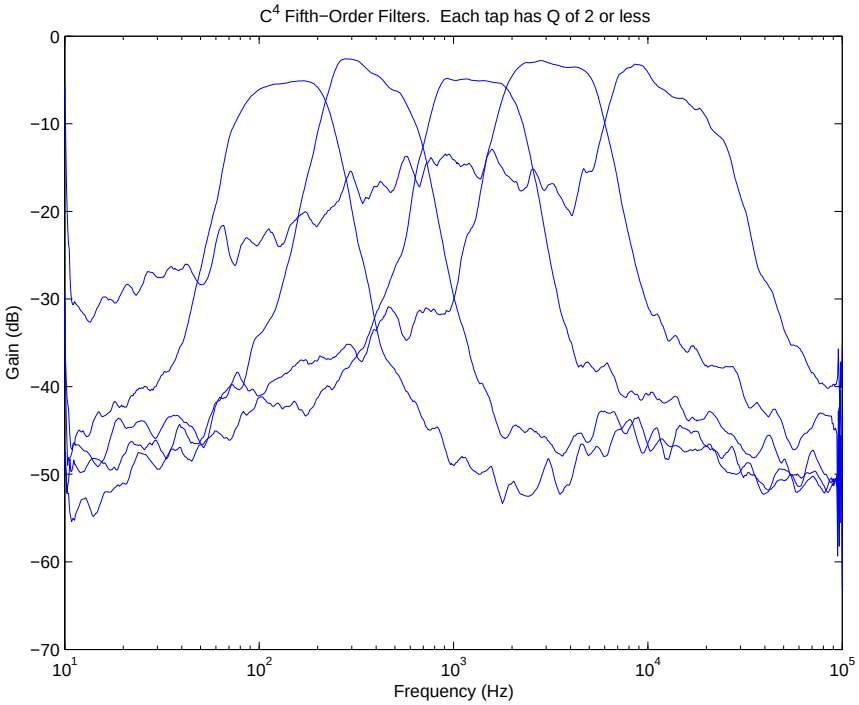


Figure 46. C^4 fifth-order filter bank response data.

a predefined filter bank. However, to expand the spectrum decomposition to any arbitrary range of frequencies, one must proved a means to set the corner frequencies to a set of values after fabrication.

This can be accomplished by using programmable Bias Generators. These cells integrate programmable elements to allow the spread of the bias values and the scale between values to vary based on the programmed value in an analog memory element.

The advantages to this method are that all bands will be spaced across the frequency of interest without doing much programming. However, the cost for this easy of use is extra circuitry required to provide the spreading function of the bias currents.

In this chapter we have shown the design parameters which affect frequency response, gain and linearity of a 5 transistor filter element. We have shown primarily experimental results, along with some simulation, to verify the tunability and linearity of the device. Taking all of these into account, this ultimately results in a low-power and compact filter which can be easily tuned over 6 orders of magnitude in frequencies with maximum Q factors that can explicitly designed into a single stage. Using these design parameters, one can begin to design higher-order C^4 filter banks, with applications in numerous signal processing applications. In this paper we have shown the design parameters which affect frequency response, gain and linearity of a 5 transistor filter element. We have shown primarily experimental results, along with some simulation, to verify the tunability and linearity of the device. Taking all of these into account, this ultimately results in a low-power and compact filter which can be easily tuned over 6 orders of magnitude in frequencies with maximum Q factors that can explicitly designed into a single stage. Using these design parameters, one can begin to design higher-order C^4 filter banks, with applications in numerous signal processing applications.

CHAPTER 4

ANALOG SIGNAL PROCESSING BLOCKS

In this chapter we present the dynamics of several floating-gate computational buildings blocks and accompanying experimental measurements. These blocks include programmable signal spectrum decomposition, peak detection, vector matrix multiplication for performaing analog frequency transforms, and specifically in our case cepstrum processing. This chapter will also cover analog blocks that perform a trigonometric distance measure and diffuser elements for wave propagation networks. Using these blocks we are able to begin developing analog architectures as front-ends for larger digital or analog speech processing and recognition systems.

4.1 Frequency Decomposition

Transforms are widely used and in a general sense can refer to any type of conversion operation that may be applied to a signal. Some common transforms include: the Fourier Transform (FT), Discrete Fourier Transform (DFT), Wavelet transform (WT), etc. Block transforms have a fundamental issue of space-frequency localization. This is particularly important to discuss when dealing with windowing issues associated with performing the transform. The longer the window, the better the frequency resolution; the shorter the window, the better the time resolution. The tradeoff is unavoidable and is limited by the *Fourier uncertainty principle* which also limits these circuits even though they operate on continuous signals.

The wavelet transform is a tool that cuts up data or functions or operators into different frequency components, and then studies each component with a resolution matched to its scale. Wavelets provide a tool for time-frequency localization of a signal evolving in time (e.g., the amplitude of the pressure on an eardrum, for acoustical applications). Our ear uses a wavelet-like transform when analyzing sound, at least in the very first stage. Given

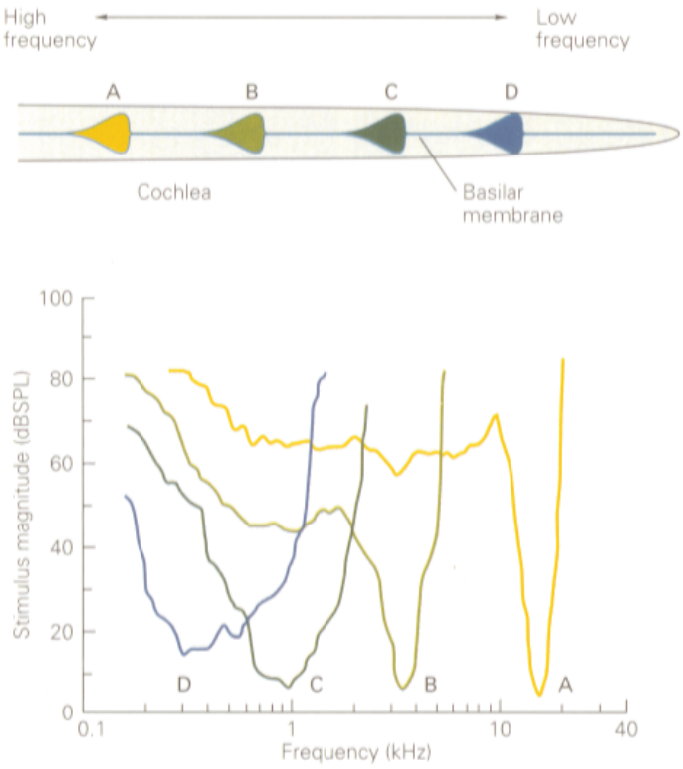


Figure 47. Frequency response of the cochlea.

a signal f(t), one is interested in its frequency content locally in time. The standard Fourier transform also gives a representation of the frequency content of f, but information concerning time-localization of, e.g., high frequency bursts cannot be read off easily from the Fourier transform. Time-localization can be achieved by first windowing the signal.

In some filterbank configurations, particularly when using octave bands, one obtains a discrete-time wavelet series. Such a configuration has been popular in signal processing less for its mathematical properties than because an octave band or logarithmic spectrum is more natural for certain applications such as audio compression since it emulates the hearing process. Such an octave-band filter ban an be used, under certain conditions, to generate wavelet bases, as shown by Daubechies [71].

This section explains the concept of spectral decomposition using analog programmable

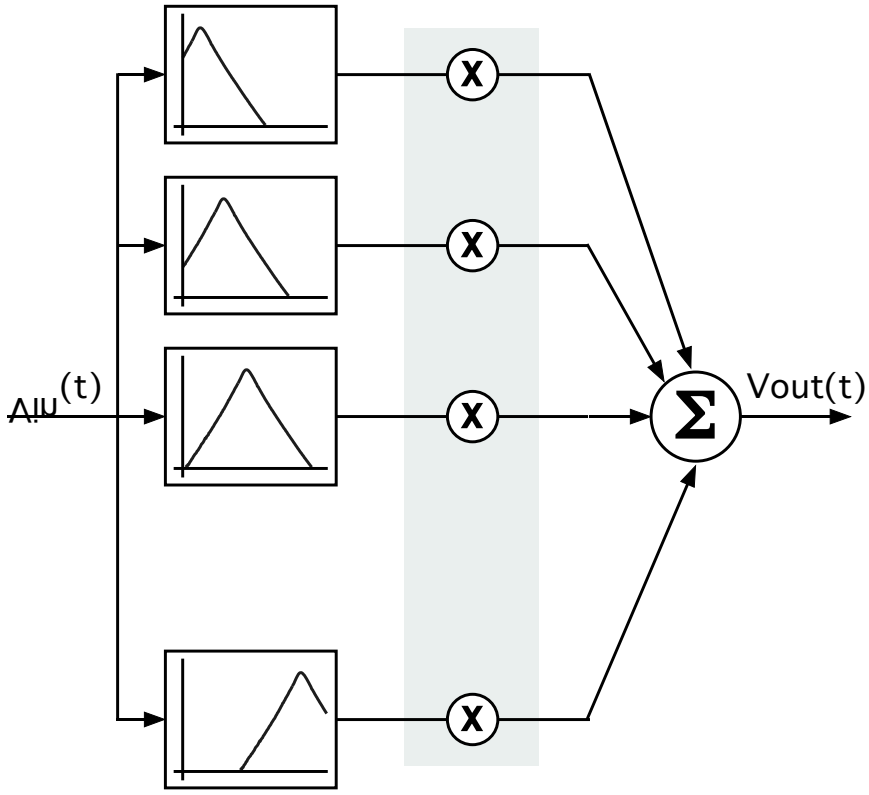


Figure 48. Filter bank approach to decomposition and similarities to classic way of thinking about FFTs.

filter bank elements. In this section we present an analog filterbank operation which is analogous to a Continuous-time Fourier Transform. Given specific implementation properties of the filterbank, similarities to the wavelet transform are also possible. These elements operate at extremely low power and lend themselves well to low-power real-time audio signal processing. This bank of analog bandpass filters can be placed before an analog-to-digital converter to perform a frequency decomposition of an audio signal and also allow for analog preprocessing before being transformed into the digital domain. The individual filter taps are easily be tuneable to any given frequency and bandwidth because capacitively coupled current conveyers (C^4s) are used with floating-gate transistors as biasing elements. Offsets and mismatches within the circuit elements are shown to be inconsequential because they can be accounted for and programmed out.

In order to build a useful analog frequency-decomposition block, a bank of bandpass

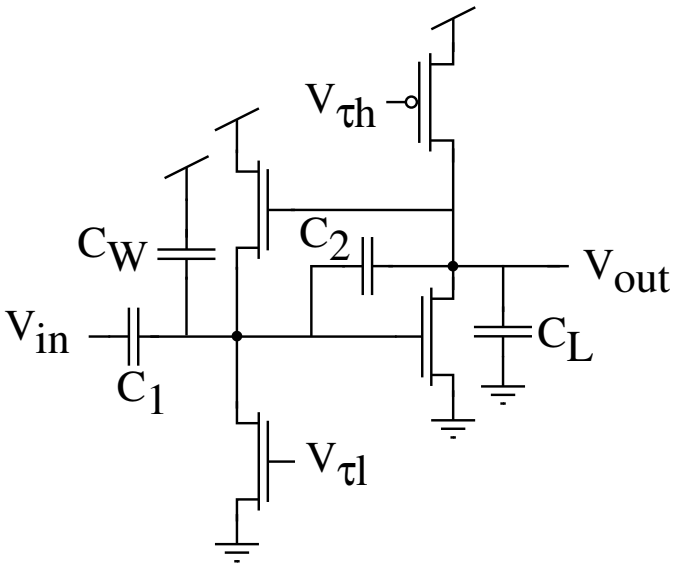


Figure 49. Schematic of a single C^4 structure. The capacitors model all explicit and parasitic capacitances in the signal path of the circuit.

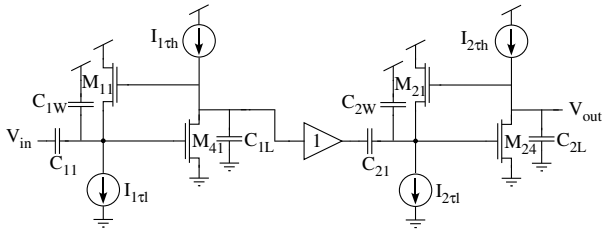


Figure 50. Floating-gate version of a capacitively coupled current conveyer second-order section (C^4SOS).

filters has to be created in which the basic bandpass filter element must be relatively small so that it can be placed in an array, the bandpass element must be easily tunable so that multiple elements can cover the entire auditory spectrum, and the center frequencies of the bandpass elements must follow an orderly spacing. Typically, for frequency decomposistion, exponential spacing is desired. Having a moderate amount of resonance ($Q \approx 30$) is also desirable for better isolation of the center frequency. The rest of this chapter discusses how we have gone about designing and building this type of programmable bandpass array.

A similar operation to Sub-band coding, one can view the classical transform operations as a bank of filters. Given a specific filter bank decomposition, we would like to explain,

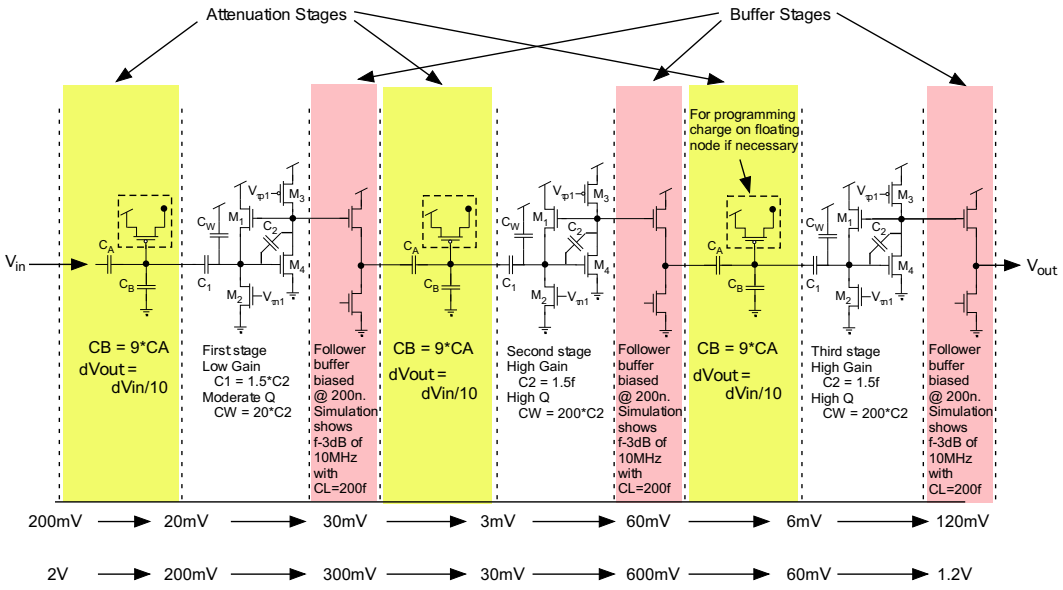


Figure 51. This figure illustrates the method used in the most recent filter bank chip with attenuation on the input as well as a buffer on the output. This allows for the most input swing for improved linearity, as well as isolating the filter from capacitance shifts on later stages.

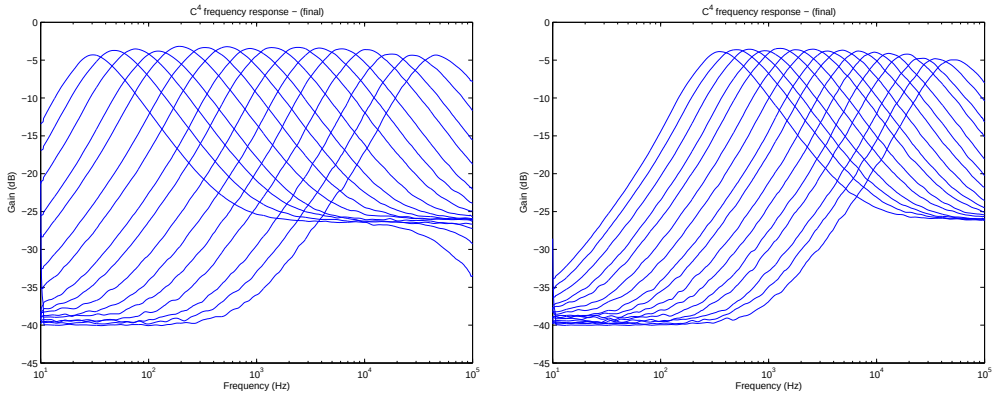
analytically, the specific transformation taking place. Because we are using constant-Q filterbanks, these operations are similar to wavelet transforms. The difference here is that for our designs, the maximum achievable Q is predesigned, while the resulting Q of each filter depends on the placement of the high and low corners, which are both programmable. In order to simplify the comparison with wavelets, we will describe all filterbanks as if they have the same Q, which is required for the comparison.

Using these continuos-time filter banks for signal decomposition is possible. The filtering operation for each core filter, then using the base structure to develop higher-order filters, will have the following characteristics

$$H(s) = \frac{s(\frac{\omega_o}{Q}) * H_{BP}}{s^2 + s(\frac{\omega_o}{Q}) + \omega_o^2} \tag{30}$$

where ω_o is the center frequency and H_{BP} is the pass band gain of the filter.

A filter bank of this type does not allow for perfect reconstruction because some information will be lost during the filtering operation, where even within the pass band of the filters, there will be a corresponding ripple between filters. However, this filtering operation



(a) (b)

Figure 52. First-order C^4 filter bank response data. a.) C4 filter corners programmed to have a Q=2 with the high frequency starting at 64kHz and all the other corners falling out from there. b.) C4 filter corners programmed to have a Q=3 with the high frequency starting at 64kHz and all the other corners falling out from there.

is applicable in many places. The key question in these applications is how much overlap is reasonable between filters, which leads to the underlying design characteristics for the filters themselves such as Quality factor or ”Q” and pass band gain.

The operation of this element is covered in much more detail in Chapter 3. For audio signal we are interested in using this element in the range from 100Hz up to 100kHz. An array of 32 C^4s was fabricated with a $0.5\mu m$ process available through MOSIS. While the bandpass filter elements can be programmed to any desired center frequency spacing and bandwidth, we will use the example of exponentially spaced center frequencies with narrow bandwidths and moderate Qs as this is a highly advantageous configuration in audio signal processing. This type of configuration closely models the biology of the human cochela [cite] and it lends itself to allow subbands of frequency to be independently manipulated since there is a real-time frequency decomposition occurring. Figure 52 shows the frequency response of 16 of the 32 filter taps. Only the output from the first stage (a single C^4) is shown. These filters were programmed so that the $I_{\tau l}$s and $I_{\tau h}$s for each filter tap had exponentially spaced currents that were programmed within 95% of their desired values.

By programming the filter bank to have exponentially spaced center frequencies with

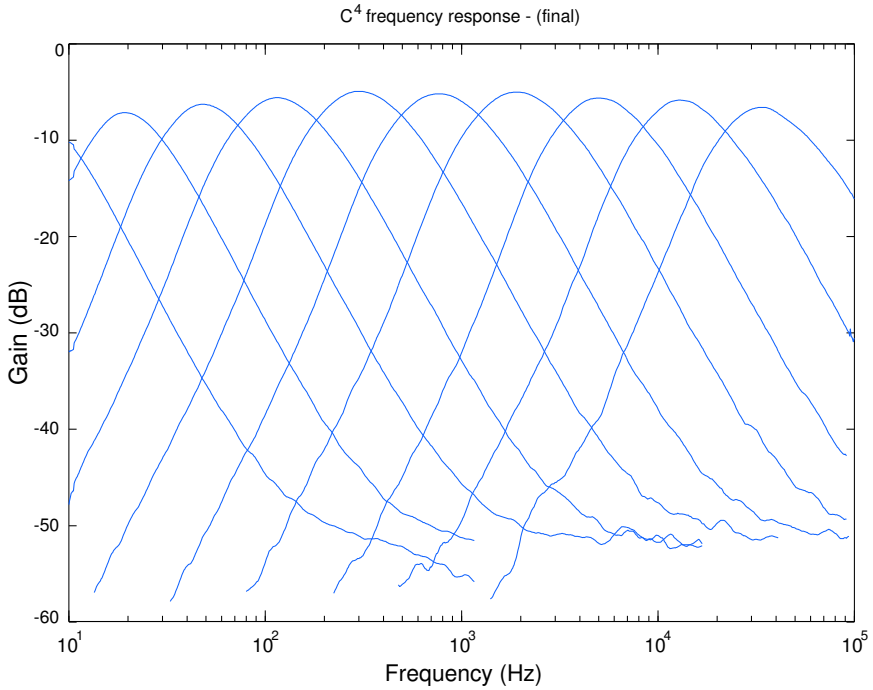


Figure 53. Second-order C^4 array programmed with a Q of 1.

narrow bandwidths and moderate resonance, an input signal is broken down into its respective frequency components. This type of filter bank is extremely useful in signal processing applications in which the specific algorithm is easier to perform on individual subbands of the input signal, as is often done in auditory processing

Useful applications

1) Cepstrum Encoding ASR

2) Noise Reduction

3) Biological Hearing Models

Show outputs of speech data.

In this chapter we have shown the design parameters which affect frequency response, gain and linearity of a 5 transistor filter element. We have shown primarily experimental results, along with some simulation, to verify the tunability and linearity of the device. Taking all of these into account, this ultimately results in a low-power and compact filter

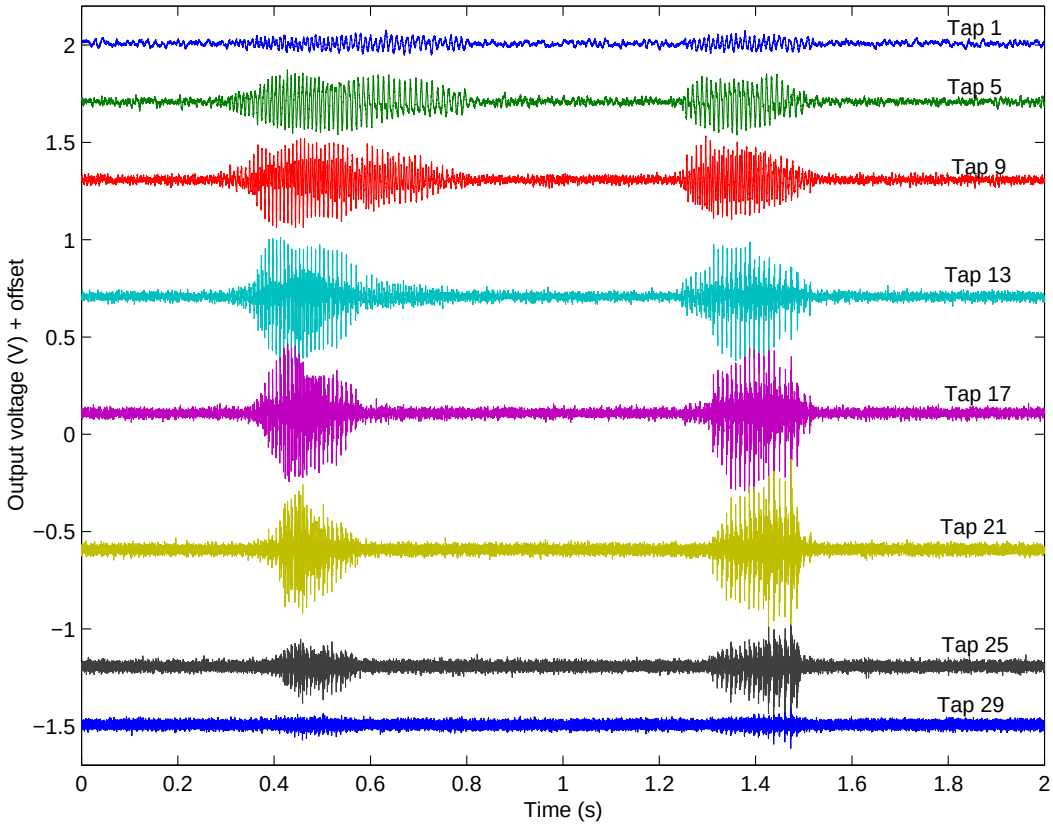
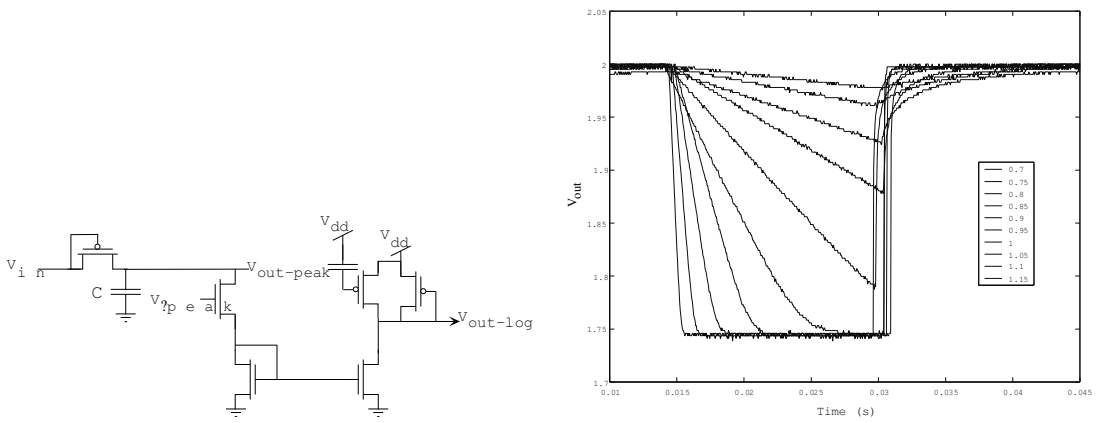


Figure 54. Output from a speech signal with two segments of spoken words. The outputs for the following filterbanks is shown.

which can be easily tuned over 6 orders of magnitude in frequencies with maximum Q factors that can explicitly designed into a single stage. Using these design parameters, one can begin to design higher-order C^4 filter banks, with applications in numerous signal processing applications. In this paper we have shown the design parameters which affect frequency response, gain and linearity of a 5 transistor filter element. We have shown primarily experimental results, along with some simulation, to verify the tunability and linearity of the device. Taking all of these into account, this ultimately results in a low-power and compact filter which can be easily tuned over 6 orders of magnitude in frequencies with maximum Q factors that can explicitly designed into a single stage. Using these design parameters, one can begin to design higher-order C^4 filter banks, with applications in numerous signal processing applications.

4.2 Amplitude Detection



(a) (b)

Figure 55. a.) The initial peak detector circuit using a classical diode for the input. This circuit suffered from decreased input range because of the initial diode drop, as well as very large offsets. b.) Initial step response data showing the adjustable time constant of the circuit.

The magnitude of each spectrum passes through a peak detector stage to produce a constant magnitude output. This magnitude is similar to taking the power spectrum density or real spectrum of an input signal. At this point, phase information is unchanged, however the frequency response of the peak detectors must be programmed to it's respective frequency band.

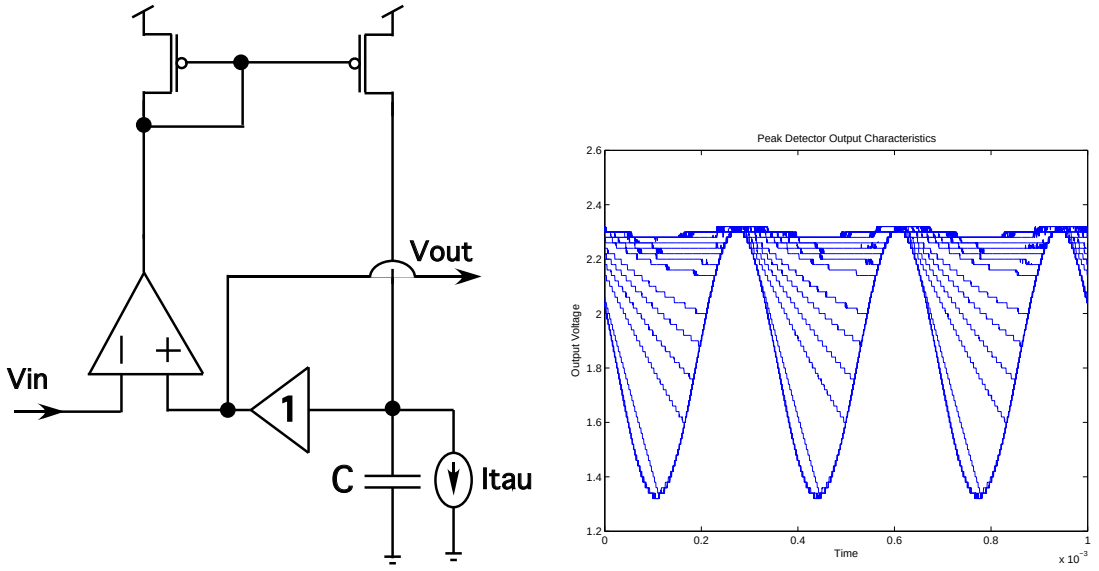
The initial circuit is shown in Figure 55. The input diode has the following current relationship

$$I_D = I_0 e^{\frac{\kappa Vin - V_2}{U_T}} \tag{31}$$

This current $I = V_2/R$, where R is the total resistance of the bias transistor and diode. The floating-gate transistor on the output provides an offset current to set the DC output voltage. The output diode has the a logarithmic relationship to current.

$$\Delta V_{out} = -\frac{U_T}{\kappa} ln(\frac{I - I_{offset}}{I_0}) + v_{dd}. \tag{32}$$

Each peak detector has an individually programmable corner frequency. Because the output magnitude is continuous, this allows us to capture additional high frequency content



(a) (b)

Figure 56. a.) The new peak detector circuit using a feedback to track to signal peaks. This circuit does not suffer from decreased input range due diode drops. There are also no system offsets outside of the amplifiers which are typically small, and minimized by the overall loopgain due to the feedback configuration. b.) Tracking data of the new peak detector as the bias current is changed.

within each band. The peak detector programming blocks are isolated similarly to the C^4s. The entire bank is treated as a single row and within that row the individual elements are accessed by column. Control circuitry on the rows and columns ensures isolation.

A new design for the floating-gate minimum and peak detector is shown in Figures 57 and 56 respectively. The circuit uses an high-gain feedback loop to track the maximum and minimum of input signals. The circuit is also capable of locking to the maximum/minimum, however for many signals with a varying amplitude, such as speech, it is advantageous for the circuit to track these changes. The tracking speed for an instantenous increase in the speech signal depends on the bandwidth of the feedback amplifier which is in the 10MHz range, well above the frequencies of interest. However, as signal energy within a particular band decreases the tracking speed of the envelope detector is related to a linear decay relationship based on a bias current and capacitor.

$$\tau = \frac{I}{C} \tag{33}$$

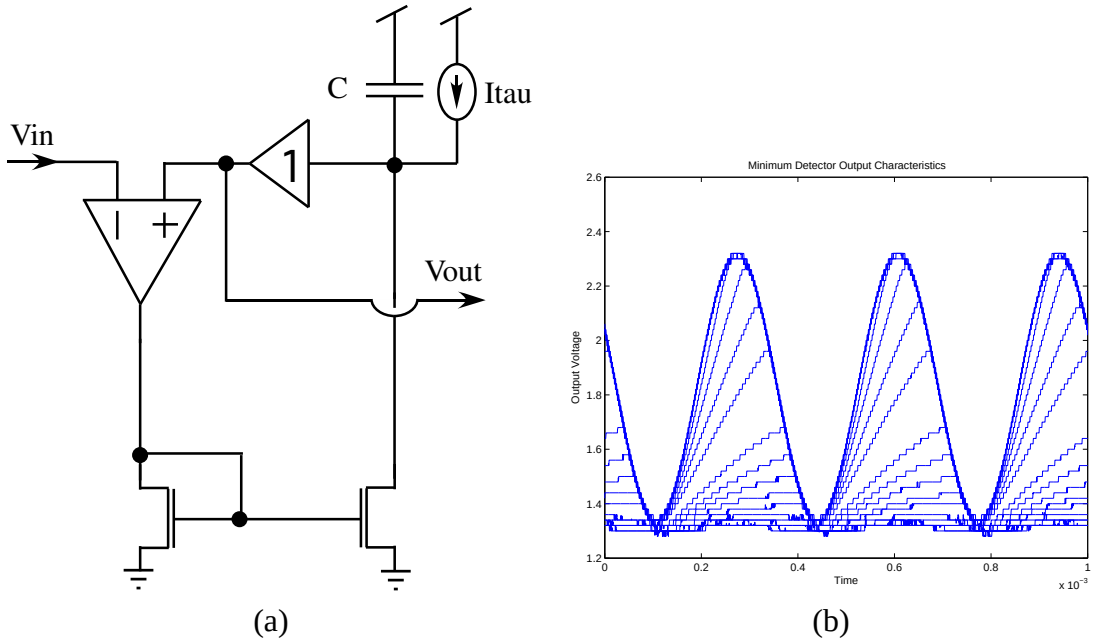


Figure 57. a.) The new min detector circuit using a feedback to track to signal peaks. This circuit does not suffer from decreased input range due diode drops. There are also no system offsets outside of the amplifiers which are typically small, and minimized by the overall loopgain due to the feedback configuration. b.) Tracking data of the new min detector as the bias current is changed.

The initial design used a 100fF capacitor which resulted in bias current on the order of pico-amps to track signals in the kilohertz range. The latest version doubled the capacitor to increase the low-end useable range of the circuit. This circuit has the added advantage of being easily modified to operate as a minimum detector. The combined circuit performs a signal envelope function which is later used as our magnitude output from the filter banks.

4.3 Linear transconductance

The initial stages of the system operate in voltage-mode on both inputs and output and up to this stage, that has worked fine. However, at this point we would like to convert out frequency dependent voltage signals into a current to be weighted during the transform operation. This section explores our phases of development towards our transconductance element, or more simply a voltage-to-current convertor, designed utilizing floating-gate inputs. The element displays a linear transconductance response over a limited range range

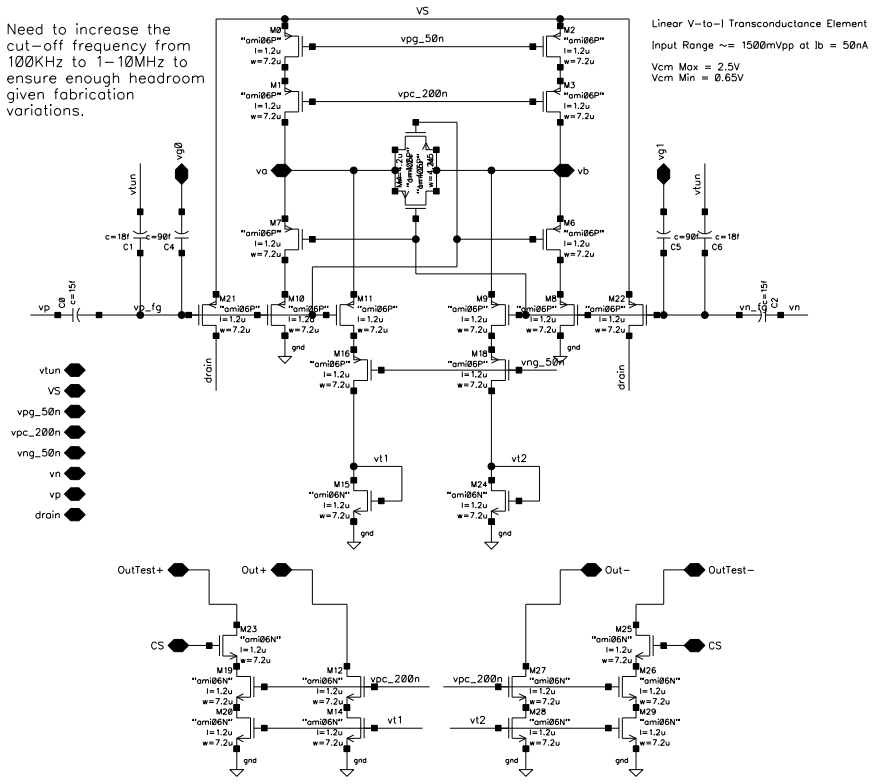


Figure 58. Circuit diagram illustrating a floating-gate implementation of a linearized transconductance stage.

of voltage inputs. To increase the input range we can implement capacitive source attenuation. Capacitively coupling the inputs does more than simply attenuate the input signal, it also allows us to use our programmable memory concept to these cells as well in order to cancel any offsets at the input. Experimental data is presented from circuits fabricated on a 0.5μm nwell CMOS process available through MOSIS.

Initially one would think to use a fully-linear transconductance stage at this point, which will have many benefits such as linearity, obviously, along with other benefits such as classic and well know design methodologies [34]. These techniques, however, utilize the square-law behavior of the transistors in above-threshold which increases the power dissipation of these stages. Again, assuming the final implementation will be to use an array of these elements, power is of significant concern. Along these same lines there are also

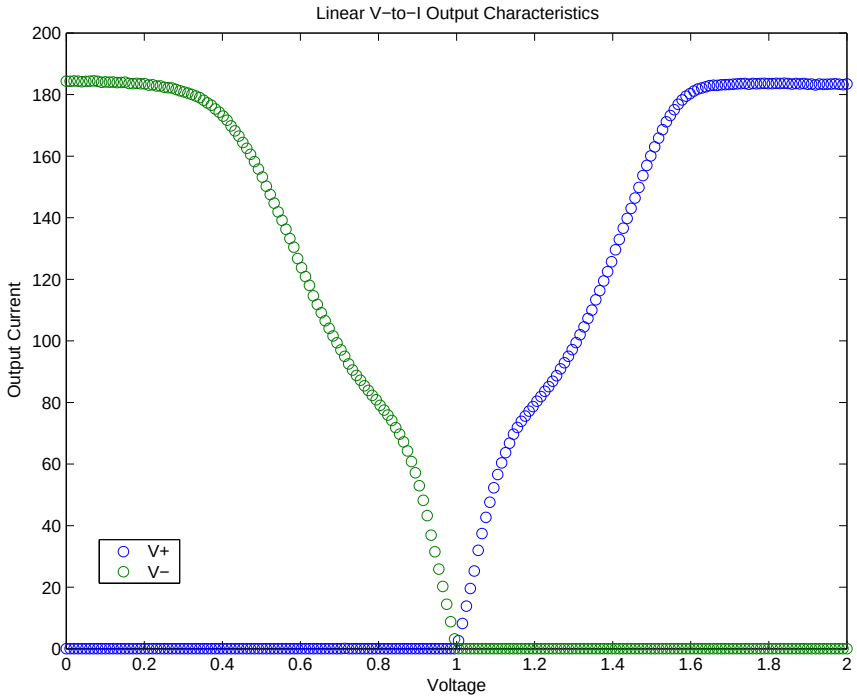
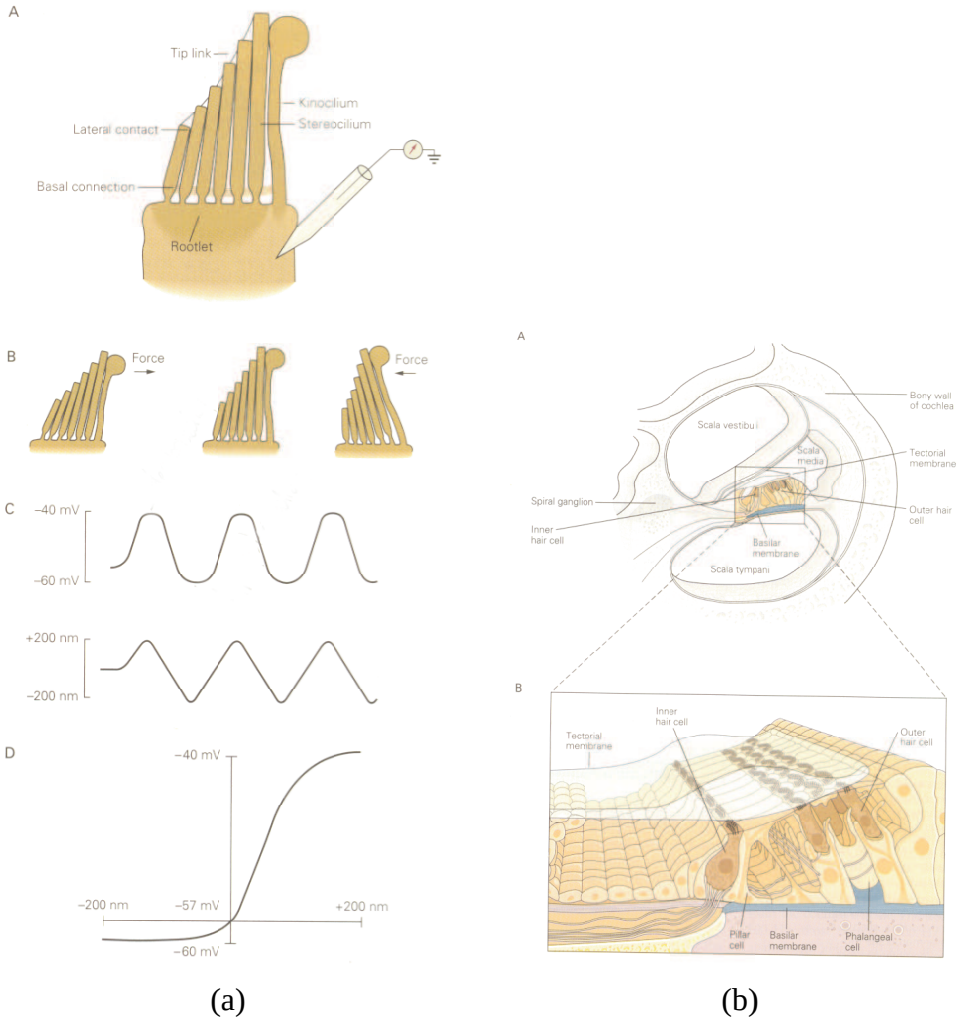


Figure 59. Response of Linear Transconductance stage used to change the differential input voltage into a single-ended output current.

techniques that can operate with the devices in sub-threshold using what is called source-degeneration, and specifically for these circuits, the technique bears a striking resemblance to diffuser concepts [76].

An initial design was done combining the classic and low-power transconducance linearization techniques. The circuit shown in Figure 58 uses a technique of splitting the differential input current on each leg such that both input signal (V+ and V-) are input on each leg. Along with this, I included the cross-connected transistors at the source of the devices to include additional source-degeneration. The final goal being a circuit with a linear voltage-to-current relationship over a fairly wide input voltage. Results also shown in Figure 59 show a final input range of 500mV, but a less than ideal linear response. There is a significant drop in the transconductance around zero differential input voltage, due primarily to the transistors used for source-degeneration. Ideally we would want these devices connected to a voltage relative to the common-mode input voltage, but for sake of



(a) (b)

Figure 60. A. A schematic drawing of a hair cell with a recording electrode inserted into its cytoplasm. B. Application of a mechanical force to the hair bundle deflects this elastic structure. C. When the top of a hair bundle is displaced back and firth by a stimulus probe (lower trace), the opening and closing of mechanically sensitive channels produces an oscillatory receptor potential (upper trace). D. The sigmoidal relation between hair-bundle deflection (abscissa) and receptor potential (ordinate) is a stimulated hair cell. Figure taken from [?].

area these voltages were connected to the input. This technique proved to be less than ideal for the final application.

Along a different train of thought, looking more at the biological structures involved in auditory transduction including the outer-ear, cochlea and inner hair cells, there are some characteristics of this portion of the auditory path that prove to be useful. Of particular

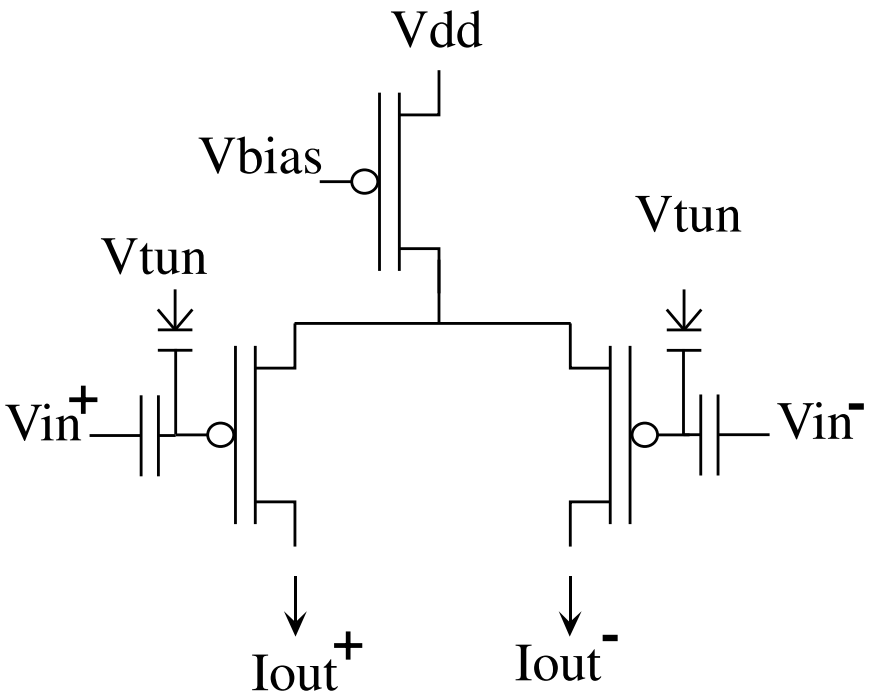
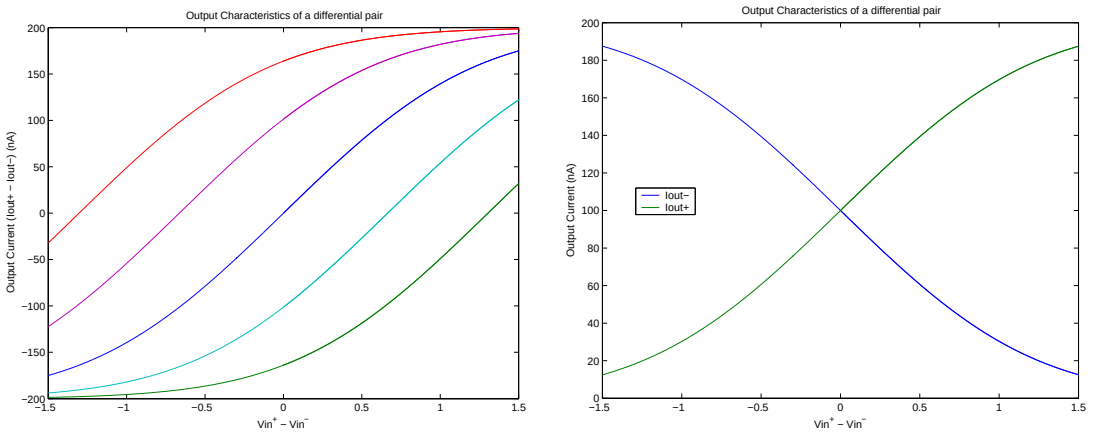


Figure 61. Circuit diagram illustrating a floating-gate implementation of a normal differential input stage.

interest is the hair cells as shown in Figure 60. The hair cells transduce mechanical motion on the cochlea to electrical signals processed by the brain. The transductance curves are sigmoidal in nature, meaning they saturate beyond a given input magnitude. This is caused by two physical characteristics of the hair cells, limited range of motion and limited neurotransmitter response. Because of this, there are many other structures involved in the very complex auditory pathway to compensate for these limiting factors and to maintain the sensitivity of this portion of the auditory system, but we are focussing primarily on the hair cells as they relate to our method of transducing voltages to currents at this critical stage. Using this sigmoidal response as a basis, we can simply implement this stage as a differential pair.

The very well-known differential input pair, as shown in Figure 61, has two huge benefits for this system, very compact and also low-power, based on whatever value we choose,



(a) (b)

Figure 62. Response of the floating-gate differential input pair showing improved input linear range, in addition to the desired saturation effects as also seen in the hair cell in the auditory pathway. The floating-gates also allow the zero-bias point to be shifted by changing the charge on the floating-gate.

the obvious choice being sub-threshold for consistency. The differential input pair equation

$$I_{out}^{+} - I_{out}^{-} = I_{bias} \tanh(\frac{\kappa(V_{in}^{+} - V_{in}^{-})}{2U_T}) \tag{34}$$

gives us the same sigmoidal response as an inner hair-cell as shown in Figure **??**. The input linear range for this circuit is typically on the order of +/- 50mV. In order to increase the input linear range, we take advantage of our floating-gate techniques. This circuit is shown in Figure 61 and using capacitive attenuation, we are able to increase the input linear range to +/- 250mV, or 500mV peak-to-peak as shown in Figure 62.

Although the differential-pair has a linear input voltage of up to 500mV peak-to-peak, this circuit actively takes advantage of the saturation effects from the differential-pair. This saturation performs a limiting function on the inputs, but still preserves the sensitivity for small inputs, which is ideal for this circuit. The current outputs are also ideal for input to the next stage which is the vector-matrix multiplier block.

4.4 Programmable Multiplier

The analog differential multiplier shown in Figure 63 is using voltage input signals, scaling this signal by the stored weight and generating an output current. The equation governing

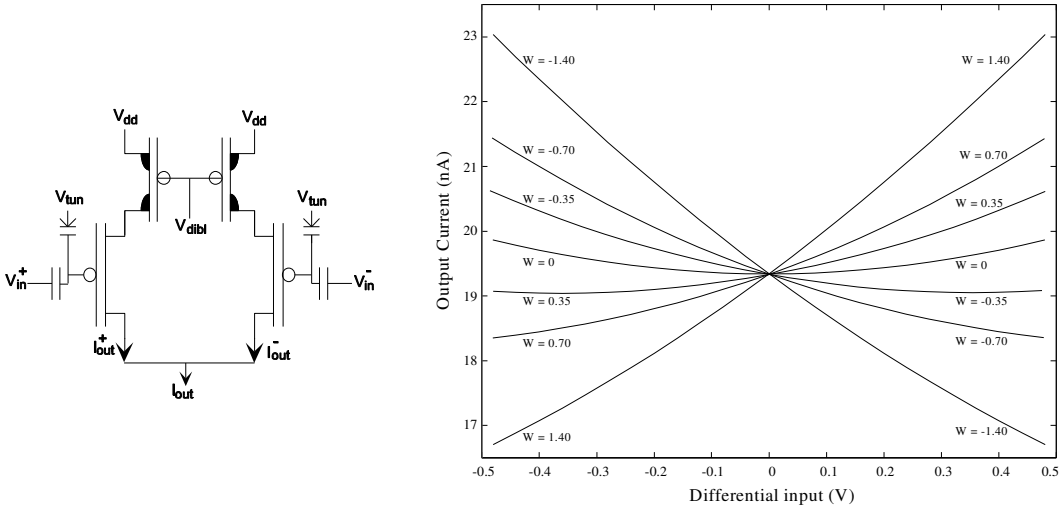


Figure 63. Voltage-mode floating-gate multiplier.

its operation is as follows:

$$I_{Out} = I_{SO}(W^+ + W^-)cosh(\Delta V_{in}/V_y)$$
$$-I_{SO}(W^+ - W^-)sinh(\Delta V_{in}/V_y) \quad (35)$$

The linearized case including only the first order terms simplifies to:

$$I_{Out} = I_{SO}(W^+ + W^-) + I_{SO}(W^+ - W^-)(\Delta V_{in}/V_y)$$

Where the two weights W^+ and W^- are programmable floating gate voltage values. These values can be programmed to any arbitrary value, but for operations involving spectrum decomposition and transforms, these values are programmed to cosine scale factors. Their differential operation requires each pair to have a DC bias voltage and from there, the cosine values are scaled around this bias current. A sample application would be for each row to act as a 32-tap DCT basis vector. Similar vectors are programmed into each row and typically they will be set to produce a wavelet type of decomposition by scaling each basis vector frequency by a constant factor.

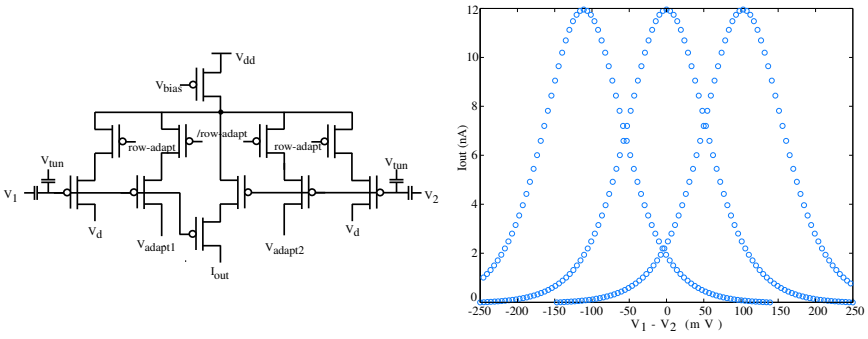


Figure 64. The programmable bump element and the corresponding output given three different programmed offset values.

4.5 Programmable bump element

Figure 65 shows the circuit and measured data for a single node in the VQ classifier array. Each cell in the array compares the value of that column's input to the value it has memorized; the output current flows out of the V_{out} node. This circuit is a variation on the bump circuit [8], which compares the two inputs to this circuit; this cell returns a high value if the two values match (minimal difference). The circuit performs a continuous distance computation along a particular input coordinate:

$$I_{out_k} = \frac{I_b}{2} sech^2\left(\frac{\kappa(V_k - V_{mk})}{2U_T}\right), \tag{36}$$

where V_k is the differential input voltage, V_{mk} is the resulting stored voltage representing the ideal mean value for this particular element, κ is the coupling from gate to surface potential, and $U_T = kT/q$ is the thermal voltage. This system outputs a measure of the similarity; therefore, the outputs of all the elements can be added (by KCL) and the largest output is the vector with the maximum similarity. The sum of these current outputs are sent through a Winner-Take-All circuit that outputs the N largest results, where N can be 1 or more [41].

Programmable elements [21] are used at the inputs to store and subtract off the each cell's mean value. Setting the floating-gate charge establishes the mean value as well as eliminating any mismatch between the two-transistor pairs. Setting the size of the input capacitor as well as other capacitor elements around the floating-gate sets the linear range

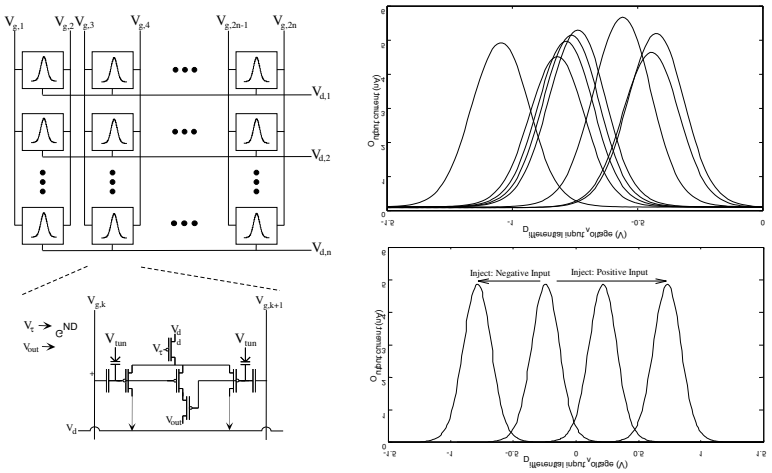


Figure 65. Programmable VQ using floating-gate circuits. We reconfigure the VQ circuit so that it fits within the standard floating-gate programming architecture and algorithms [40]. Capacitors with the additional arrows is our symbol for a tunnelling junction, which is a MOS capacitor that can also remove charge from the floating-gate node. We show experimental results after programming eight cells to different offset voltages. The difference in the bump peaks is due to mismatch in the MOSFET transistors. We reset the floating-gate charge using electron tunnelling, and program positive or negative offsets using hot-electron injection. If we inject the floating-gate associated with the positive input terminal, then we increase the offset. If we inject the floating-gate associated with the negative input terminal, then we decrease the offset.

of the circuit, and therefore sets the width of the *bump* element. We increase the floating-gate charge (remove electrons) by electron tunnelling, and decrease the floating-gate charge (add electrons) by hot-electron injection.

4.6 Programmable diffuser element

This section presents a programmable diffuser using floating-gate circuits. We present the dynamics of classical diffuser circuits and show the differences between the classical and the programmable case. Programmable diffusers offer many advantages including removing individual element mismatch and also giving the user the ability to reconfigure the overall system behavior. Programmable diffusers can be used for spreading, just as in a resistive network, and they can also be used to create applications based on wave propagation.

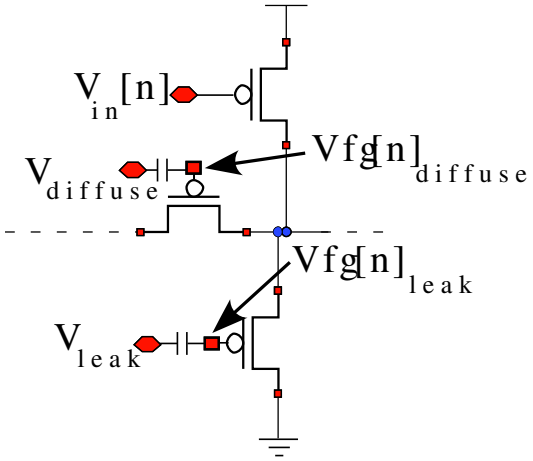


Figure 66. (a.) A "classical" diffuser element. $V_{diffuse}$ controls the lateral current propagation, while V_{leak} controls the vertical current propagation. The input gate voltage to the leak and diffusive elements are typically a single voltage so the conductance of all elements is equal. (b.) A single floating-gate diffuser cell. The floating-gate voltage of each element will control the individual conductances. The individual conductances can be programmed to any arbitrary value. Some interesting affects can be observed when voltages are equal or increase/decrease down the length of the diffuser line. Using floating-gates to set the voltages of each node allows this circuit to remain compact.

Numerous applications require temporal filtering in either one or two dimensions. Simple resistive grids have been designed to solve this problem. They offered a low-pass filtering operation on the output signal and spread the input response across multiple elements.

Resistors are very expensive when used within integrated circuit technology. Keeping this in mind, the transition to more dense networks of elements have been designed that use subthreshold MOSFETs as the diffusive elements. This offered huge savings in terms of chip area and also gave way to more elaborate diffusive networks. These elements are compact and programmable. The network characteristics are dependent upon the relationship between various floating-gate voltages and can be programmed to exhibit multiple behaviors.

A diffuser network is a current mode implementation of a one dimensional resistive network. Classic diffuser circuits have been used for many years in computational networks that require varying degrees of spreading. It can also be used to perform local spatial averaging, which can be used to improve signal-to-noise ratio or obtain a local reference to which signals can be compared. One example application involved a resistive array for

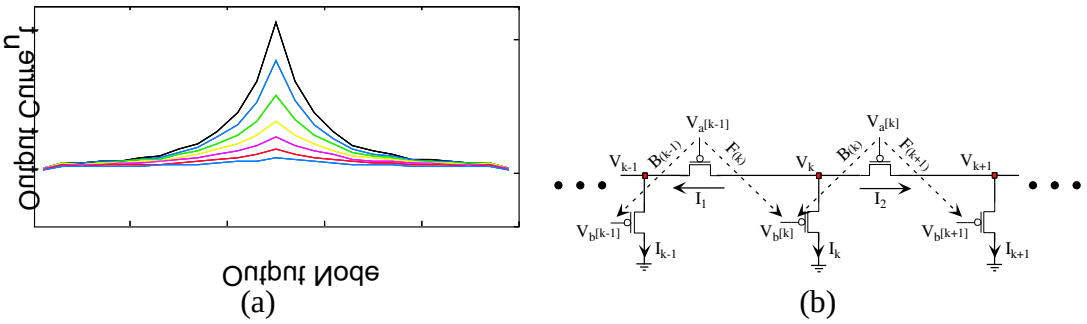


Figure 67. (a.)Output current for a "classical" diffuser element. The change in output current is due to the change in input voltage V_{in}. The input occurs at the center of the array, node V_k. $V_{diffuse}$ and V_{leak} were held constant. (b.) Diffuser circuit with node voltages labelled. The difference equations are calculated at each node and determine the direction of propagation down the diffuser line.

vision chips [57]. They have also been used to model layers of visual cortex when input signals are distributed across multiple elements. This property acts as a low-pass filter by softening any abrupt changes on the output signals.

Standard current mode diffusers spread current from a given input node. These types of networks can be realized using either passive or active components. Classic diffusers will have a constant conductance and a constant leak term at each node [57, 4]. Diffusers made with active elements, such as transistors, can vary the conductance by changing the gate voltage. This change is typically global. $V_{diffuse}$ controls the lateral current propagation, while V_{leak} controls the vertical current propagation. By using floating-gate transistors, we are able to vary each conductance individually. The behavior of the diffuser circuit can be described as

$$I(x) = \frac{I_{in}}{2L} e^{-x/L} \tag{37}$$

where

$$I_{in} = I(x), \text{ where } x = 0 \tag{38}$$

with $x = 0$ representing the input node, and L is the space constant. There is also a capacitance at each node that results in low-pass filtering behavior, which is nonlinear.

Implementing a diffusor network using floating-gate elements to set the conductances

of the individual diffuser and leak elements has two distinct advantages. The first is cancelling mismatch when floating-gate diffusers are used in the classical diffuser with a global constant conductance. By programming the floating-gate charge to a desired value and thus setting a desired conductance, offset is limited by the accuracy with which you can program a floating-gate element [22, 80]. The second is the ability to change the individual conductance of each element. The relative floating-gate voltages can be programmed such that the conductance of adjacent nodes increase in one direction and decrease in the other, thus introducing a propagation gradient.

Figure 67 shows a standard diffuser circuit with node voltages labelled. Beginning at node V_k we define the node currents as if the nodes of the diffusive elements to the left and right are sources with currents I_1 and I_2 respectively. The connected node of the leak element is the source terminal of the element with current I_k leaving the node. There is also an input current from a synchronous input which is labelled I_{in_k} and a capacitance (C) at each node which produces a voltage delay to an input current pulse at the node.

Summing these node currents gives us:

$$I_{in_k} = I_1 + I_k + I_2 + C\frac{dV_k}{dt} \tag{39}$$

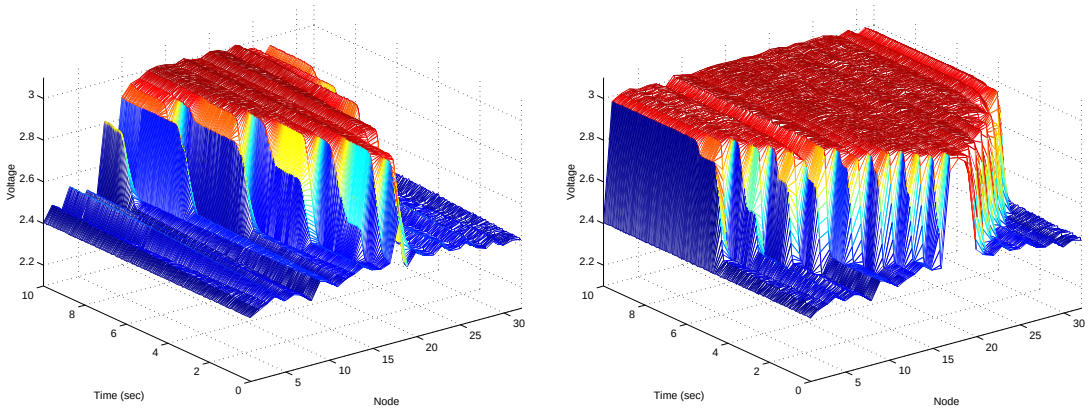
Rearranging and grouping terms we get

$$\begin{aligned} I_{in_k} &= -(\frac{C \cdot U_T}{I_k})\frac{dI_k}{dt} - I_{k-1}e^{-\kappa/U_T(V_{a_{k-1}}-V_{b_{k-1}})} \\ &- I_{k+1}e^{-\kappa/U_T(V_{a_k}-V_{b_{k+1}})} \\ &+ I_k + I_k e^{-\kappa/U_T(V_{a_k}-V_{b_k})} \\ &+ I_k e^{-\kappa/U_T(V_{a_{k-1}}-V_{b_k})} \end{aligned}$$

Defining a backward propagation term and a forward propagation term respectively, we get

$$B(k) = e^{-\kappa/U_T(V_{a_k}-V_{b_k})}, \;\; F(k) = e^{-\kappa/U_T(V_{a_{k-1}}-V_{b_k})}$$

B(k) represents the backward gate voltage differences between the current diffusive element



(a) (b)

Figure 68. Diffuser line output voltage for each element vs. time with symmetrical conductances and input applied to the center. (a.) Diffuser line output voltage for each element vs. time. There is a single input applied at the center of the diffuser line. The time delay at each node is dependent upon the conductance of each diffuser block and also the magnitude of the input pulse. (b.) Diffuser line output voltage for each element vs. time with a larger input pulse.

and the previous leak element and F(k) the forward gate voltage differences between the previous diffusive element and the current leak element, see Figure 67(b).

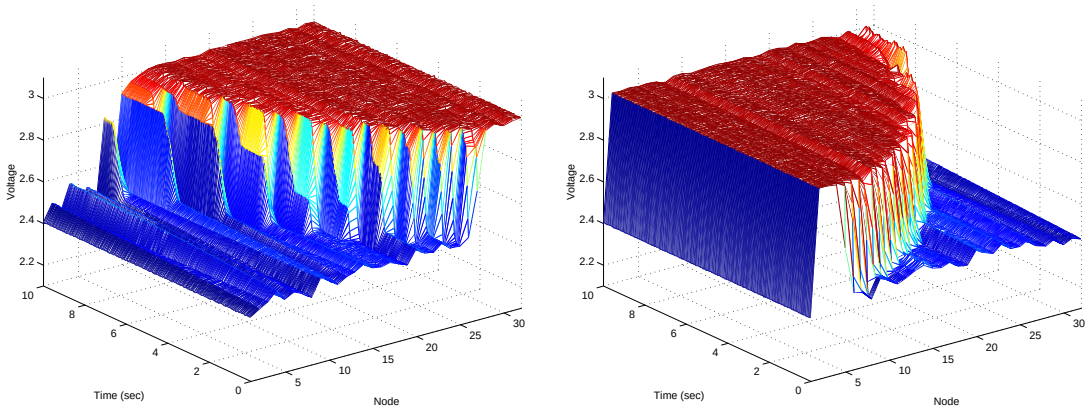
Using these simplifications and substituting for $F(k)$ and $B(k)$, we can further relate the spacial delay to a time delay for each node, and move the delays to a unit delay in order to rewrite everything in terms of the Z-Transform in space. This results in the following equation

$$\begin{aligned} I_{in_k} &= -\left(\frac{C \cdot U_T}{I_k}\right)\frac{dI_k}{dt} \\ &+ (1 + B(k) + F(k))I_k \\ &- B(k)I_k z^{-1} - F(k)I_k z \end{aligned} \tag{40}$$

From here we raise the question of relating the equation with components that are discrete in position, to wave propagation for a continuous case.

The Tailor series expansion for the discrete form of the second-derivative, we get the following equation:

$$zY(z) - 2Y(z) + z^{-1}Y(z) = \frac{d^2y}{dx^2} \tag{41}$$



(a) (b)

Figure 69. Diffuser line output voltage for each element vs. time with symmetrical conductances and input applied at each end. (a.) Diffuser line output voltage for each element vs. time for a single input applied to the right of the diffuser line. (b.) Diffuser line output voltage for each element vs. time with a single input applied to the left of the diffuser line. The output voltage drop looks linear for short diffusion distances but is actually exponential. The current drop is linear, therefore the voltage drop is exponential. This can easily be seen with a drop shown across the entire diffuser line.

This differential equation is capable of supporting diffusive and wave-propagating solutions. From (40) we are able to get three distinct behaviors from the network.

(1) Diffusive Case: $F = B$

$$B[z - 3 + z^{-1}]$$

$$-B + B\underbrace{[z - 3 + z^{-1}]}_{\frac{d^2}{dx^2}}$$

(2) Propagating Case 1: $F \gg B$ and F is equal

$$[Fz - (1 + F)]$$

$$-1 + F[\underbrace{[z - 1]}_{\frac{d}{dx}}$$

(3) Propagating Case 2: $F \ll B$ and $\forall$ elements

$$-(1 + B) + Bz^{-1}$$

$$-1 + B\underbrace{[z^{-1} - 1]}_{-\frac{d}{dx}}$$

Programmable diffusers can approximate both a propagating wave

$$\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x} \tag{42}$$

and a diffusive wave

$$\frac{\partial I}{\partial t} = \frac{\partial^2 I}{\partial x^2} \tag{43}$$

Figures 68 and 69 show experimental results with for a system with all of the conductances set to be equal which is the diffusive case. Any input on a single node will spread symmetrically to it's neighbors on both sides. In order to create wave propagation the floating-gate elements can be programmed with a voltage gradient in the desired direction. The system can propagate faster than seconds, but we show these speeds to show how slowly the propagation can occur, which is important for many sensory (speech, vision) computations / classifications.

CHAPTER 5

ANALOG SIGNAL PROCESSING SYSTEMS

5.1 Auditory Feature Extraction

This section presents a novel approach to programmable spectrum decomposition, analog frequency transforms, and spectrum compaction. This system can act as a feature extraction front-end for larger digital or analog speech processing systems.

From the general model of speech production, speech is the convolution of an excitation sequence, which is a pseudorandom sequence, with an impulse response of the vocal system [9]. Extracting the excitation or vocal tract response from a speech signal is non-trivial because they are combined by convolution:

$$s(n) = e(n) * \theta(n), \tag{44}$$

where $s(n)$ is the speech signal, $e(n)$ is the excitation, and $\theta(n)$ is the vocal tract response. Many applications require some knowledge or at least an estimate of either the pseudorandom excitation or the vocal tract response or both.

There are many different methods of feature extraction. Different feature extraction techniques can be used depending on wether the desired features relate to auditory properties or general signal properties. Examples of this are cepstrum processing, linear predictive coding, and zero-crossing processing. Cepstrum has its roots in the characteristics of the human auditory system in that a speech signal can be modelled as a random excitation sequence convolved with the transfer characteristics of the physical articulatory components such as larynx, vocal chords, tongue, teeth and lips. LPC processing is dependent upon correlations in the auditory sequence and statistics of previous sequence values. Other techniques look at the zero crossing of a signal to extract useful information and base their processing techniques on properties of biological components of the auditory system such the hair cell[44, 45] and cochlea [54, 84]. Other implementations look at modelling the

overall auditory system [83, 48]. There may be other techniques but these are some of the most common. Regardless of the feature extraction technique, there are benefits and hurdles when using any of them and one must understand the information that is obtained from any of the coding techniques. For our purposes, cepstrum processing was chosen because of its parallelism with a filter bank implementation, its resulting feature set, and that it follows closely with what has been implemented using DSPs.

Feature extraction techniques that have been implemented either in DSP or analog/mixed-signal IC have different limiting factors. In DSP, limitations include power vs. computational cycles, if power is a concern, otherwise computations are limited to real-time implementation constraints. In the analog domain design constraints include power, but typically orders of magnitude less than digital circuitry; chip or die area; signal factors such as dynamic range and signal-to-noise ratio; or general mismatch in devices due to process variance. For our purposes one can follow good design procedures to minimize limiting factors such as signal-to-noise or process variances, but die area, for the algorithms we are proposing, is a critical limiting factor. In order to perform real-time matrix multiplications, frequency transforms or high-order signal processing tasks in analog circuitry would require huge die-area to accomodate sample-and-hold circuitry.

The speech signal is modelled by a convolution of a random excitation sequence e(n) with the vocal tract v(n). The excitation sequence can be modelled as primarily high-frequency information, while the vocal tract changes at a much lower frequency. The Cepstrum essentially separates the high-frequency from the low-frequency data by performing a log-compression on the spectrum information.

Several kinds of spectral matching measures have been studied. In particular, spectral matching measures based on linear predictive coding (LPC), which seem to be superior to matching measures based on the filter bank or on the fast Fourier transform (FFT), have been shown to be very effective in speech recognition, speech analysis, speech synthesis, and speech coding.

The LPC *cepstrum distance* (CEP) was introduced by Atal(1974). Itakura(1975) introduced an efficient measure called the Itakura distance or the likelihood ratio measure and showed its usefulness for spoken word recognition. Moreover, he clarifies its meaning as a spectral matching measure in the frequency domain. Gray and Markel (1976) summarized the LPC-based measures, together with the Cosh measure, from the standpoint of matching in the frequency domain and successfully applied these measures to vector quantization.

The LPC cepstrum, the Itakura distance, and the Cosh measure show the same characteristics in matching between similar spectra (Gray and Markel, 1976). According to our evaluation of these measures by spoken word recognition (Shikano and Sugiyama, 1982) and vowel verification (Shikano and Kohda, 1980), the Itakura distance is somewhat inferior to the LPC cepstrum distance and the Cosh measure because of its asymmetrical spectral matching in the frequency domain. The LPC cepstrum distance and the Cosh measure demonstrate quite similar performances for word recognition and vowel recognition. These measures, however, lack two characteristics of frequency domain spectral matching. One is a local peak-weighted matching like formant matching. The other is a frequency axis weighting such as provided by Mel scaling.

Cepstral analysis is a special case within a general class of methods known as "homomorphic" signal processing [61]. The cepstrum is able to resolve the two convolved pieces of the speech, $e(n)$ and $\theta(n)$, into two additive components, that can then be separated or analyzed using spectral (cepstral) analysis. In general, the spectrum $\Theta(\omega)$ of the vocal tract response is assumed to be smooth with only slow changes as a function of frequency; the spectrum $E(\omega)$ of the excitation signal is assumed to have rapid variations as a function of frequency. When cepstrum analysis is used, the energy from $\log|E(\omega)|$ will be mapped to high values of n and the energy from $\log|\Theta(\omega)|$ will be mapped to low values of n. Therefore, the two signals, $c_e(n)$ and $c_\theta(n)$, will largely occupy different parts of the frequency axis and can be analyzed as separate entities.

A common variation on the real cepstrum is the Mel-cepstrum which combines the

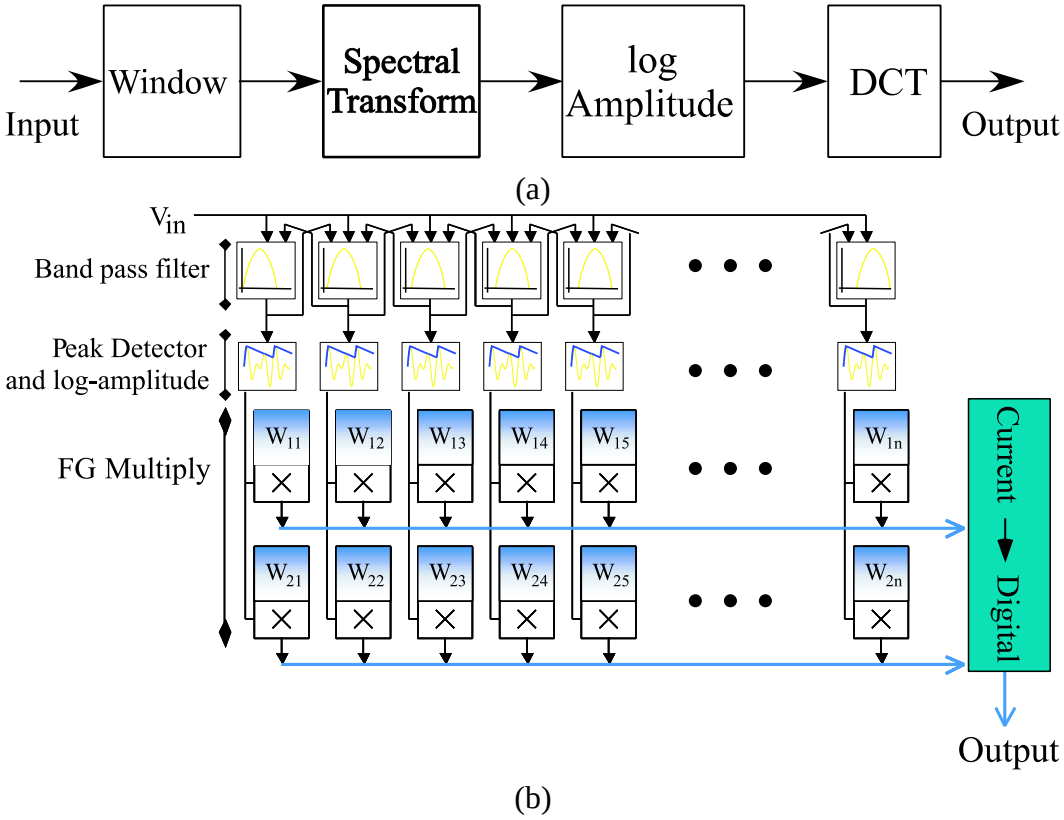


Figure 70. Respective digital and analog cepstrum implementations. a.) The traditional cepstrum computation which is performed in digital circuitry. b.) Floating-gate system to perform cepstrum front-end computation for speech processing systems. The system contains 32 frequency taps that can be spaced arbitrarily by programming the corner frequencies for the bandpass filter banks. The peak detectors provide a power spectrum of the input signal for any given time slice. Outputs are generated by summing the output currents for each row through KCL.

output of the $\log|S(\omega)|$ into critical band energies prior to performing the inverse DCT. The Mel-cepstrum has certain qualities that make it useful for speech recognition. First, the critical band filtering stage produces an output that is very similar to that observed at early stages of human auditory perception. Second, the final DCT serves to decorrelate the critical band energies for improved automatic pattern recognition performance. The DCT also has the advantage over the Fourier transform that it yields real values. It is for these reasons that the Mel-Frequency Cepstrum is the model used in designing our analog cepstrum implementation.

Within the analog circuits domain, an analog Mel-Frequency Cepstrum is implemented beginning with a band-pass filter approach to signal decomposition. Large arrays of floating

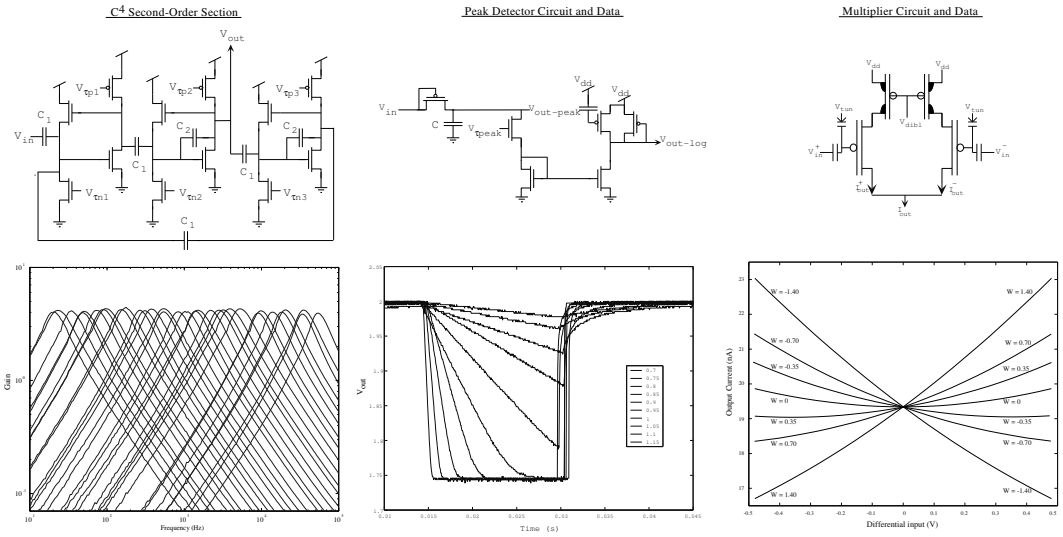


Figure 71. Analog Cepstrum building blocks. a.) Floating-gate C^4 second-order-section and its corresponding frequency response. The high and low corner frequencies can be independently tuned for each filter bank. Arbitrarily programmable corner frequencies allow these filters to be spaced linearly, octave, logarithmically or any other values desired by the user. b.) Differential floating-gate multiplier structures multiply two differential signals by constant factors that are stored on the floating gate elements. c.) Floating-gate peak detectors. The frequency response of the peak detector is controlled by a bias voltage which controls the gate of nFET M3. This element sets a constant resistance and the total R,C value shifts the high corner frequency. The frequency response is shown for different values of v_{tau}.

gate computational blocks have been developed that perform similar processing as mel-cepstrum algorithm. The processing is all done in continuous analog circuitry.

The basic building block of the cepstrum begins with a continuous spectrum decomposition similar to a Discrete-Fourier Transform (DFT). The spectrum decomposition is done using differential Capacitively Coupled Current Conveyors (C^4) second-order-section bandpass filters [17]. The spacing of the bandpass filters is arbitrary because each can be programmed to have a desired high-frequency corner and low-frequency corner [40]. These structures can also be cascaded to design higher order filters. Figure 30 shows the frequency response for the C^4 structure for 1^{st},2^{nd} and 3^{rd} order filters. For the auditory applications Q's of up to 30 are useful which would limit implementation to a minimum of a 2nd order filter structure. For simplicity only one half of the differential structure is shown in Figure 71. Programming the C^4s is handled as if each filter were two floating gate elements. The entire row is viewed as a single row and the floating-gate elements for

the high and low corner frequencies are accessed by column. Control circuitry guarantees injection isolation by latching the gate and drain voltages to the power supply for elements within C^4s not selected [39].

The magnitude of each spectrum passes through a peak detector stage to produce a constant magnitude output. This magnitude is similar to taking the power spectrum density or real spectrum of an input signal. At this point, phase information is unchanged, however the frequency response of the peak detectors must be programmed to it's respective frequency band. The circuit is shown in Figure 71. The input diode has the following current relationship:

$$I_D = I_0 e^{\frac{\kappa Vin - V_2}{U_T}} \tag{45}$$

This current $I = V_2/R$, where R is the total resistance of the bias transistor and diode. The floating-gate transistor on the output provides an offset current to set the DC output voltage. The output diode has the a logarithmic relationship to current.

$$\Delta V_{out} = -\frac{U_T}{\kappa} ln(\frac{I - I_{offset}}{I_0}) + v_{dd}. \tag{46}$$

Each peak detector has an individually programmable corner frequency. Because the output magnitude is continuous, this allows us to capture additional high frequency content within each band. The peak detector programming blocks are isolated similarly to the C^4s. The entire bank is treated as a single row and within that row the individual elements are accessed by column. Control circuitry on the rows and columns ensures isolation.

The analog differential multiplier is shown in Figure 71. The equation governing its operation is as follows:

$$\begin{aligned} I_{Out} = I_{SO}(W^+ + W^-)cosh(\Delta V_{in}/V_y) \\ -I_{SO}(W^+ - W^-)sinh(\Delta V_{in}/V_y) \end{aligned} \tag{47}$$

The linearized case including only the first order terms simplifies to:

$$I_{Out} = I_{SO}(W^+ + W^-) + I_{SO}(W^+ - W^-)(\Delta V_{in}/V_y)$$

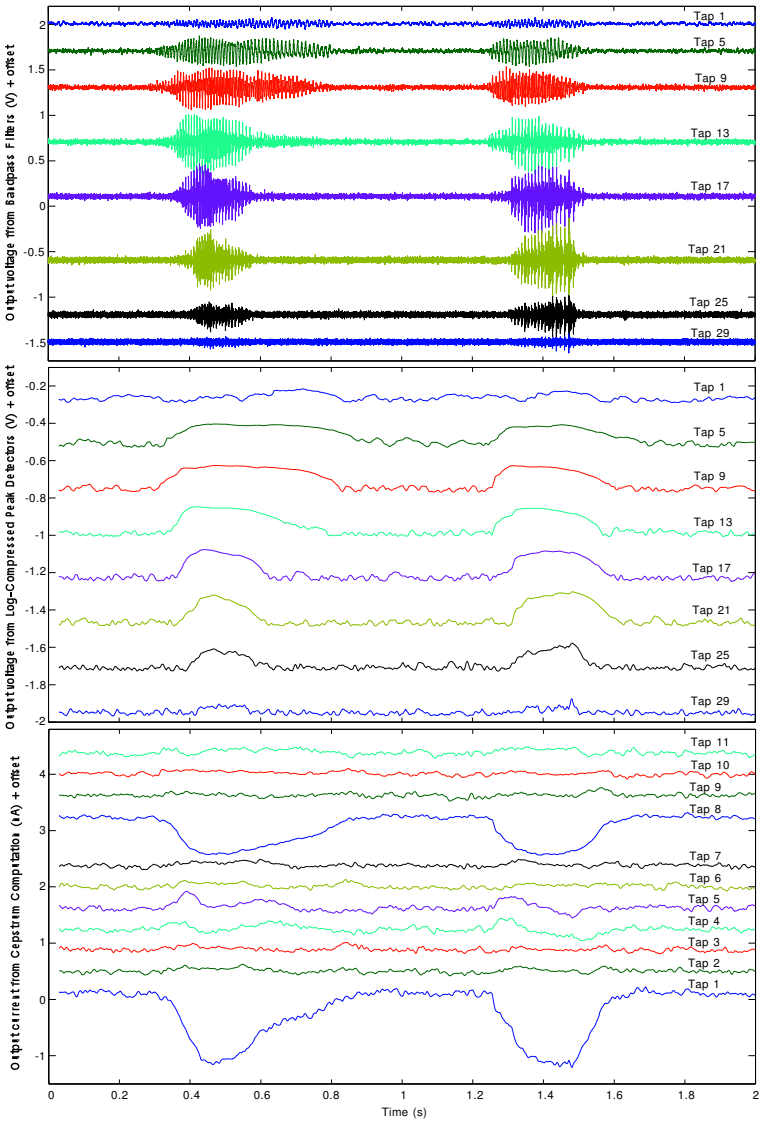


Figure 72. Cepstrum system output. The system input is a sequence of speech using a standard speech database. There are 12 continuous cepstrum coefficients calculated for this section of speech and more coefficients is only a matter of chip area since the calculation is performed in parallel analog circuits. From the graph one can see the two distinct periods of speech.

Where the two weights W^+ and W^- are programmable floating gate voltage values. These values can be programmed to any arbitrary value, but for operations involving spectrum decomposition and transforms, these values are programmed to cosine scale factors. Their differential operation requires each pair to have a DC bias voltage and from there, the cosine values are scaled around this bias current. Each row acts as a 32-tap DCT basis vector. Similar vectors are programmed into each row and typically they will be set to produce a

wavelet type of decomposition by scaling each basis vector frequency by a constant factor.

The mel-cepstrum, as used in digital signal processing (DSP) is based on a signal sampled in time and in frequency. The analog cepstrum is an approximation to the mel-cepstrum or cepstrum (depending on how the filters are defined) in which frequency is sampled but time is not. The output of each filter contains information similar to the short-time Fourier transform and can likewise be assumed to represent the product of the excitation and vocal-tract within that filter band. The primary difference here is that the DSP mel-cepstrum approximates the critical band log frequency analysis of the human ear by combining DFT bands while the analog system actually performs a critical band-like analysis on the input signal. Thus higher frequency critical band energies are effectively computed using shorter basis functions than the lower frequency bands. This is more in agreement with analysis in the human auditory system and is better suited to identifying transients.

One other difference between the analog cepstrum described herein and the real cepstrum described in Eq. (44) is that the magnitude function (inside the log) is estimated using a peak detector rather than using the true magnitude of the complex spectrum.

Chips performing an analog cepstrum have been fabricated through MOSIS. Initial results show that the analog cepstrum will be useful components within speech recognition systems. The full system is currently being tested with preliminary results are shown in Figure 72 The system output from the analog peak detector was processed using simulated multiplier cells in matlab.

5.2 Analog Pattern Recognition Blocks

5.2.1 Analog Vector Quantization

Vector quantization (VQ) is typically used in data compression and in classifying signals to symbols [72]. For example, in speech processing VQ is used to reduce the set of detectable spectrum vectors to a manageable set for later classification. The goal of VQ is to provide the simplest possible accurate description of a signal so as to minimize the subsequent

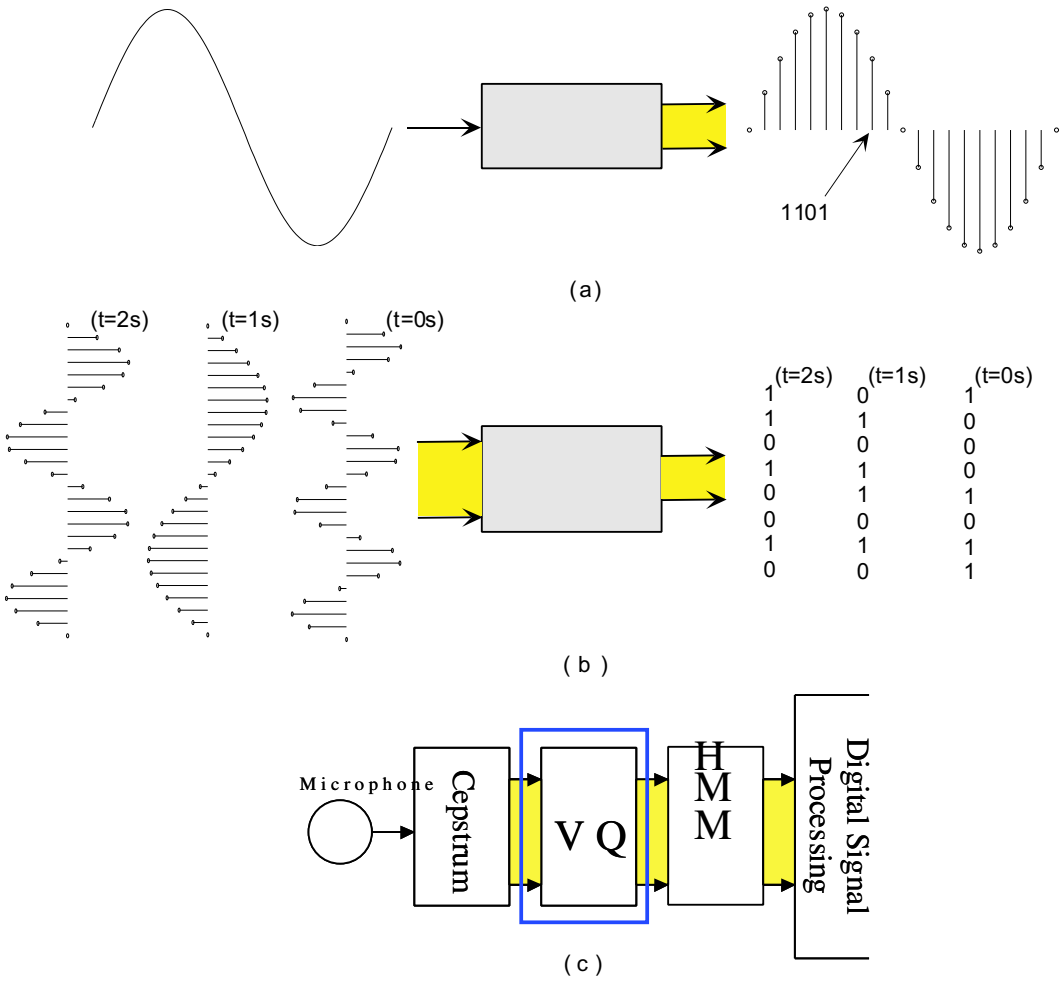


Figure 73. Overview of Vector Quantization (VQ). (a) Single-input VQ is equivalent to Analog-to-Digital Conversion. The incoming waveform is sampled and converted into digital symbols, often as a literal representation of the incoming waveform. In this diagram, we input a single channel signal at three time samples and observe that each *signal* is converted to the closest stored symbol (N-bit words for an N-bit ADC). (b) Multiple-input VQ is a multidimensional extension of the single-input VQ case. In this diagram, we input a multichannel signal at three time samples and observe that each *signal* is converted to the closest stored codeword (N-bit codewords for an N-bit VQ). (c) One application for VQ is in automatic speech recognition. In particular, a VQ block is critical in an analog IC front-end for speech recognition [75].

complexity of signal processing algorithms such as classification. VQ, like any Analog-to-Digital Conversion (ADC), is a lossy operation, but an ADC only classifies data in a single dimension, or on a vector of length one; where as a vector quantizer classifies data in an arbitrary number of dimensions, or a vector of length N. Figure 73 shows the basic concepts in VQ, as well as its application in speech recognition.

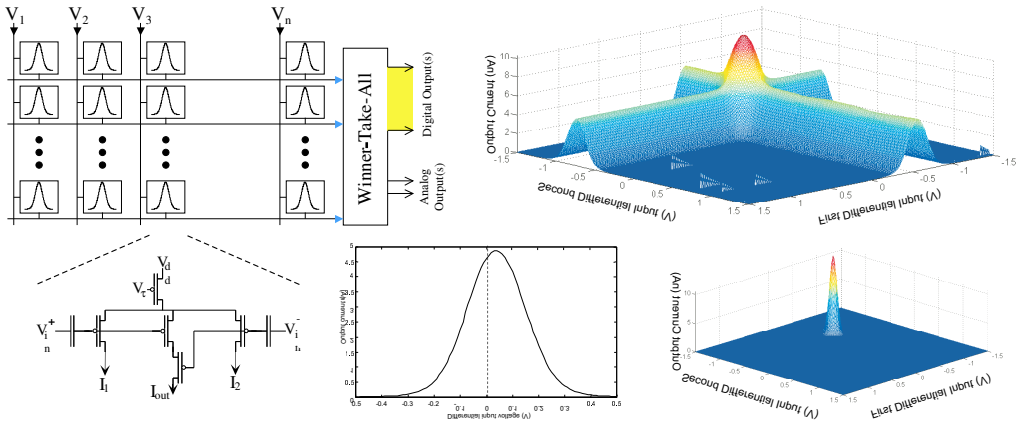


Figure 74. Basic circuit, architecture, and measurements from the VQ circuit. The core cell is built from a floating-gate bump circuit, which allows the target mean value to be stored and subtracted from the broadcasted input signal. In a 1mm x 1mm of 0.5um die area, you should be able to get roughly 1k programmable bump elements. We see that output current (experimental measurements) of the middle leg of the bump circuit reaches a maximum at its center value, and falls off exponentially as one moves from that center value. This output current is summed together with the output from other bump circuits. We also show experimental data when sweeping two inputs to see the multidimensional structure of the bump function. We get a region of high response where both bumps occur, and far away from the center, the response is nearly invariant with input voltage. Near the center of a bump in one dimension, but away from the center of the bump in the second dimension, we still see a significant response. To get an ideal VQ type bump, we would want to assume that our output is a log-encoded version of the actual output. We also show a nearly exponential expanded result using a power of 10 circuit, which is fairly easy to perform using translinear techniques (most on-chip techniques for taking the exponential of a current effectively result in a large power law).

Vector quantization can also be implemented in a programmable analog, floating-gate based system. This analog implementation of VQ provides power vs. computation savings not easily realizable in the equivalent digital VQ implementation. One can either choose to compute at higher input signal frequencies at the same power consumption, compute at the same input signal frequencies at a much lower power consumption, or a combination of these approaches. This computational efficiency is important for low-power applications, such as portable battery powered devices. We see this technology as a fundamental component in a low-power speech recognition system [75], as well as in other sensory processing applications.

A VQ system will compute how far away a particular input vector is from the desired target vectors, and pick the code vector that is *closest* to the input vector. We compute

the closest input vector by choosing an appropriate distance metric, $d(\mathbf{x}, \mathbf{m})$, between the incoming vector signal ($\mathbf{x}$) and the desired or target mean value ($\mathbf{m}$) for these signals. Two particular measures we discuss are

$$\textbf{Norm} : d(\mathbf{x}, \mathbf{m}) = ||\mathbf{x} - \mathbf{m}||_n$$

$$\textbf{Gaussian} : d(\mathbf{x}, \mathbf{m}) = e^{-||\mathbf{x}-\mathbf{m}||_2^2/(2\sigma^2)} \tag{48}$$

where n and σ is depend upon the problem and input statistics. The first approach is preferred for real-time implementation, where the second is preferred for algorithmic reasons. Previous IC implementations have used simple $||\mathbf{x}-\mathbf{m}||$ metrics for their difference functions [15, 5, 52].

The distance metric used is the following:

$$d(\mathbf{x}, \mathbf{m}) = \sum_k^n sech^2\left(\frac{x_k - m_k}{\sigma}\right) \tag{49}$$

where σ is dependant on circuit parameters. One variant uses an exponential function of this metric, which effectively turns the summation into a product, resulting in a more Gaussian-like formulation. This metric is close to the Guassian approach and easily implementable in CMOS.

5.2.2 Applications of Programmable Diffusers to HMM classifiers

In this section we show connections between dendritic processing structures and Hidden-Markov Model (HMM) decoding and a circuit topology we believe can be used to implement such a decoding structure using analog programmable diffuser networks. From an integrated circuit (IC) toward biology perspective, these simple spreading networks relate well to cable theory and are similar to biological structures such as dendrites and cortical cells. Going in the other direction, from IC to classical digital signal processing (DSP), these structure hold similarities to HMM decoders. In order to implement both structures requires a compact array of variable conductance elements. Using floating-gate transistors and allows us to get hundreds of state nodes in 1mm x 1mm of 0.5um die area and we

are able to individually vary the conductance of each diffuser element in the array, which dramatically changes the analysis of these arrays. This approach illustrates the design synergy of Neuromorphic engineering — biological systems inspire engineering design, and engineering practice inspires biological theory. This research provides a good example of how an integrated-circuit approach can bridge signal processing techniques and neural modelling.

Both techniques are based on diffusor networks, originally presented by Boahen, et. al [4]; classic diffusors have a constant conductance and a constant leak term. We use programmable diffusor networks, built from floating-gate transistors, for which we first presented experimental results elsewhere [76]. In the diffusor network, floating-gate transistors set the conductance of each element individually, thereby cancelling mismatch and allowing a desired conductance to be programmed. The result is that scaling the floating-gate voltages, particularly in a linear fashion, we can choose between classic diffusive behavior [4], described by parabolic PDEs, and forward or backward wave propagation, described by hyperbolic PDEs. Charge on each floating gate determines not only the directionality of the wave, but also its speed of propagation.

Dendritic computation is a mixture of digital and analog computing paradigms. Dendrites are a major portion of neurons. They function as the inputs to the cell and use complex connectivity to make a wide range of connection morphologies. They do not, however, merely transmit data from pre-synaptic neurons to the cell body to which they are connected; recent experimental measurements suggest 70 percent of power in the brain is supplied at dendrites. Instead, they are instrumental in performing sometimes complex computations. Since synapses (connections between cells) are made on the surface of these cells, dendrites increase the number of connections which can be made while still optimizing the cell to fit into a small space. [13]. As we previously stated, it is the dendrites that carry signals from pre-synaptic cells toward the cell body of the post-synaptic one (the cell that the dendrite is physically a part of). Whether or not the signal from the pre-synaptic

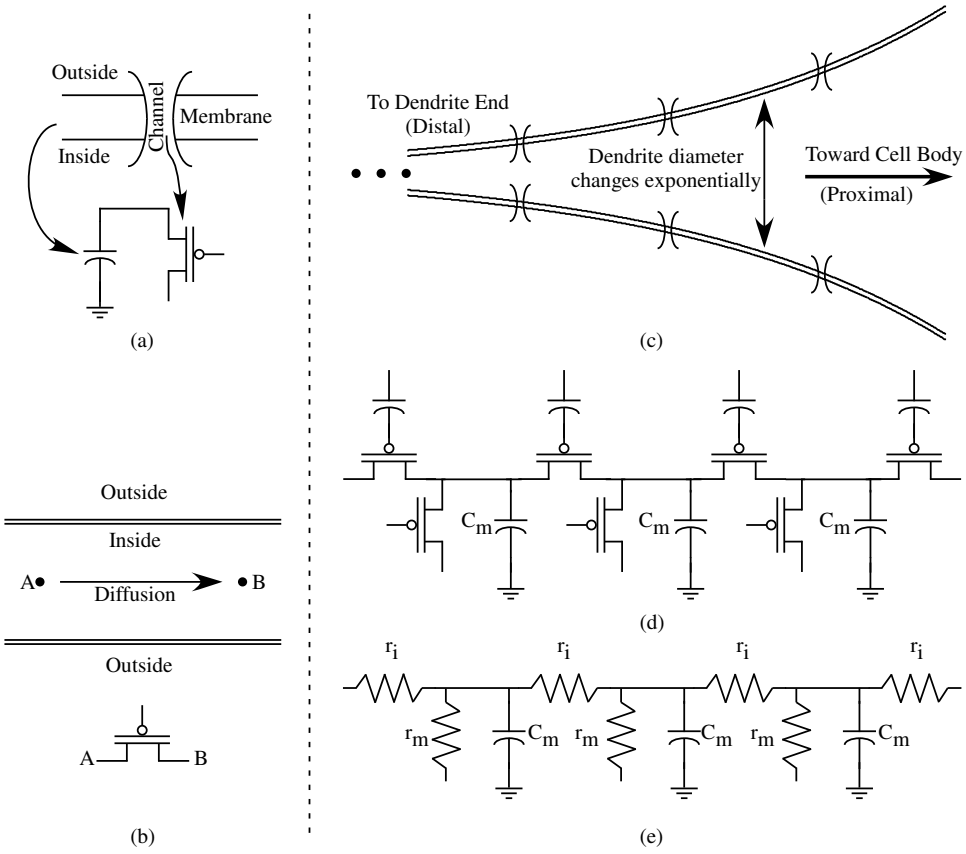


Figure 75. Dendrite figure showing component structures. (a) The membrane of a cell (including the dendrites) is comprised of a two layer lipid and channels which span both layers of the lipid. The channels selectively allow certain ions to pass across, and can be modelled by a transistor [12]. The membrane itself does a very good job of separating charge and has been classically modelled with a capacitor. (b) Inside the cell, ions spread from point A to B by means of diffusion. A subthreshold MOSFET transistor models this well as electrons spread through the channel of a subthreshold MOSFET transistor by diffusion. Putting these pieces together gives us the classic diffusor [4] which is similar to the circuit shown in (c) shows a cartoon picture of a dendrite. The diameter of a dendrite decreases exponentially as you move from the proximal to distal end (from the cell body to the end of the dendrite). (d). Here we have made the diffusive elements Floating Gate elements to greatly reduce the number of wires needed, and to facilitate programming of these voltages. (e) shows the classical view of dendrites that neurobiologists have used to model dendrites. It is similar to our view, however falls short in several areas.

neuron is able to generate a response in the post-synaptic cell depends on several factors: strength of the synapse, amplification by other excitatory signals, being suppressed by inhibitory signals, distance the signal must be transmitted, and morphology of the dendrite itself.

A Hidden Markov Model (HMM) can be viewed as a state machine in which the states

themselves are not observable, but an output, whose statistics are determined by the current state, is observable. For example, in using an HMM to model speech production the states are the desired utterance (phonemes and words) and the observations are features of the audio signal produced by the talker. The audio features are determined by the spoken word but they are randomly distributed since each time that same word is spoken it will sound a little different. For recognition problems, the goal is to estimate the underlying states of the state machine based on the observed outputs. For speech recognition, the HMM decoder takes as inputs the signal statistics or features and generates a probability of occurrence on any one of a set of speech "symbols." These "symbols" can be grouped over multiple short windows to generate larger symbols, one of which is phonemes. The ongoing input train of symbols is used to map a path through a trellis of probabilities for these larger blocks of phonemes and words [66].

It is widely accepted that Hidden-Markov Model Decoding is one of the most robust and accurate methods of speech recognition being used today. We contend that the reason for the success of HMM's, particularly in speech recognition, lies in their similarities with biological processing that is occurring at the dendrite level.

HMMs may be looked at as probabilistic state machines or some sequential processing structure. Here, we look at HMMs as propogating waves and the probabilities relate to the velocity of propogation. Previous work in this area used analog circuitry to decode the HMM states [49]; this work very clearly explained the computational paradigm for HMM classification although the circuits were not elegant implementations.

Figure 76 shows our HMM branch implementation and HMM network implementation. The transition equation is the key link between HMMs and dendritic computation structures. Because of our ability to program the floating-gates, we are able to implement the following transition equation

$$\phi_i[n] = b_i[n](\phi_i[n-1] + \phi_{i-1}[n-1]) \tag{50}$$

using the diffuser structure

$$\underbrace{\phi_i(t) - \phi_i(t-\delta)}_{capacitor(a)} = \underbrace{[b_i(t) - 1]\phi_i(t-\delta)}_{leaktransistor(b)} + \underbrace{b_i(t)\phi_i(t-\delta)}_{diffusiontransistor(c)} \tag{51}$$

The previous equation becomes

$$T\frac{\partial \phi_i(t)}{\partial t} = -(1 - 2b_i(t))\phi_i(t) + b_i(t)(\phi_{i-1}(t) - \phi_i(t)) \tag{52}$$

This shows that we can directly implement (52) using a floating-gate diffusor network, where the currents, I_i, which we define as

$$I_i = I_o e^{-\kappa V_{leak_i}/U_T} e^{\kappa V_i/U_T},$$

to represent the probability along that path.

In discrete HMM state decoding, a decoder is used with a similar structure to the true HMM system. The decoder contains states corresponding to the hidden states in the HMM system. Decoding is accomplished by selecting the most probable path through the hidden states based on the observations [9] [58].

To facilitate HMM decoding in a discrete-time system we associate a likelihood, $\phi_i(n)$ with each state i in the decoder. Then, for each time n the following steps occur:

- the outputs of the HMM system are observed and the probability that observed output was produced by state i of the HMM system is estimated as $b_i(n)$;
- for each state i the likelihood is updated based on $b_i(n)$ and the cumulative previous probabilities represented in the likelihood of all states that can transition to state i and the probability of the respective transitions.

For the stereotypical speech production HMM the likelihood update equation is

$$\phi_i(n) = b_i(n)\left((1 - a_i)\phi_i(n-1) + a_{i-1}\phi_{i-1}(n-1)\right). \tag{53}$$

[65] For the case that the transition probabilities are equal (a common assumption in speech recognition systems) the a_k terms can be neglected (i.e. the a_i and $1 - a_i$ terms are both set

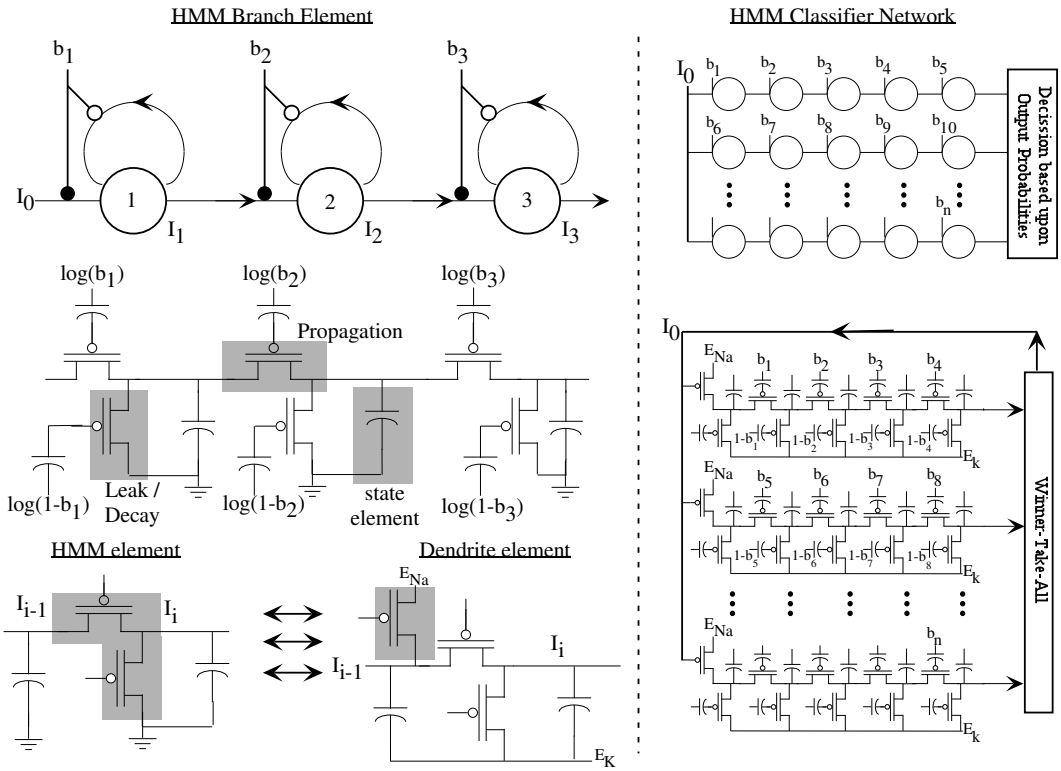


Figure 76. Circuit design for the HMM branch element as well as the corresponding HMM classifier network. Our branch element design is based upon diffusor elements to perform the classical HMM calculation. In this framework, each branch element exhibits wave propagation. We can build these branch elements into an array for classification. In a practical implementation, we need to choose the largest *useful* result, which in practice is a WTA circuit, where only a subset of winning outputs are real outputs. These real outputs reset the HMM function. At the bottom of our branch element, we also show the the relationship between a dendrite cell and an HMM cell. The difference between the two approaches is the implementation of the wave propagation mechanisms and to allow corresponding inputs to enhance wave propagation. The HMM cell explicitly combines two state elements and eliminates the leakage from the target cell. The dendrite first combines the probability to the resulting state element and then uses nonlinear gain to transmit the result to the next element. The shaded areas show the transistors that are signal dependent in each case.

to unity) leaving

$$\phi_i(n) = b_i(n)\,(\phi_i(n-1) + \phi_{i-1}(n-1))\,. \tag{54}$$

In digital signal processing, viterbi is typically the process used to calculate a path weight, however a maximum negative likelihood is typically used because of computing accuracy. Taking the log also simplifies the computation from multiplication to addition [9] [58] and also increases the dynamic range of the probabilities. The log encoding compresses the range of probabilities and ensures that accuracy is preserved.

In continuous–time HMM decoding, the likelihood update becomes continuous but the states remain discrete. We define a continuous–time HMM decoder in the same spirit as the discrete–time decoder with a likelihood update equation:

$$\frac{d\Phi_i(t)}{dt} = \gamma_i \left(\lambda_{i-1}\Phi_{i-1}(t) - \lambda_i\Phi_i(t)\right) + \beta_i(t). \tag{55}$$

In Eq. 55, $\Phi_i(t)$ typically represents a log–likelihood measure for state i; $\lambda_{i-1}\Phi_{i-1}(t)$ represents the contribution from the state $i-1$ scaled by a probability of transition; $\lambda_i\Phi_i(t)$ is the likelihood that the HMM is already in state i and that it is leaving state i; $\beta_i(t)$ is the log probability that the current observation is generated by the HMM system state i; and γ_i controls the speed of the system. Conceptually, the λ terms describe the propagation of the likelihood from left to right, γ_i controls the speed of propagation, and β_i increases the likelihood in state i when the observed output matches that expected for state i in the HMM system.

The state transition behavior of continuous-time HMMs is similar to viewing multiple waves travel along parallel paths with multiple inputs occurring at different places down the path. Depending on the current state of the wave and the input, the wave will either be amplified or attenuated. Outputs taken at the end of these paths are dependent upon the coincidence of a wave phenomena and a series of input sequences.

When applying this continuous–time HMM decoder to problems where the rate of the state sequence under observation varies greatly, it may be desirable to implement multiple HMM decoders for each state sequence. For example, if this HMM decoder is applied to speech recognition then two or three HMM decoders may be used for each phoneme to optimally identify the same phoneme spoken fast or slow. The nature of the continuous–time HMM is such that moderate variations in the rate of state changes do not present a problem but large changes may defeat the method of identification using coincident waves. It should also be noted that the disadvantage of placing multiple HMM decoders for each sequence on a chip is far outweighed by the small size of each decoder.

Using programmable diffusers gives us the ability to implement Hidden Markov Model

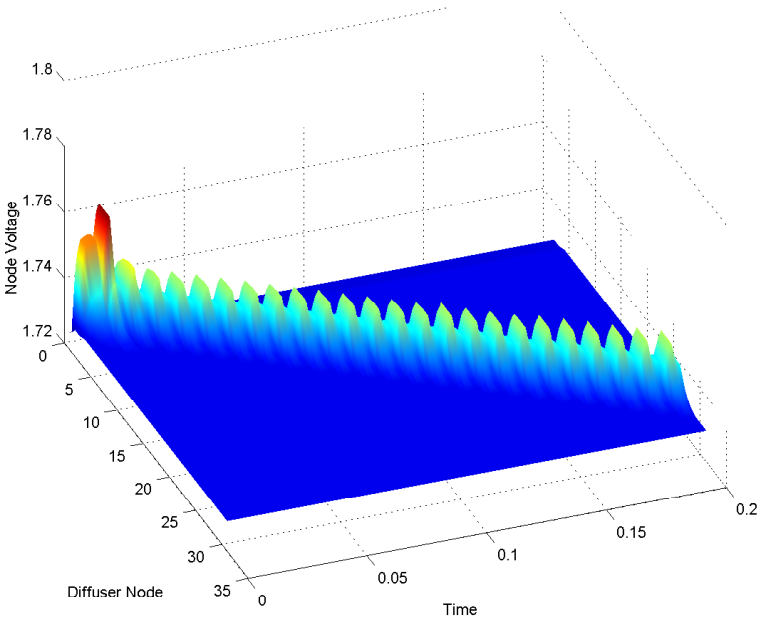


Figure 77. Results from an HMM branch with 32 nodes as a function of time. The input waveforms varied regularly with time.

decoders on an analog chip. We achieve this using passive elements for the diffuser structure, or spreading, and active elements to apply inputs at a given position and time. The core structure is a small diffuser cell contains 2-3 transistors and capacitors. The compact structure will allow us to implement very long HMM diffuser lines and/or many HMM diffuser lines. Size is important since each pattern that we desire to recognize must have an HMM decoder line associated with it [79] [49].

The core of analog HMM decoders are the programmable diffusers. We will store a wave directionality term on each floating-gate element and as a wave propagates down a particular path, these stored values determine the amount of additional driving force a wave receives as it passes through a node. Each node has a single input source and it is the coincidence of the passing wave and the input which determines the wave behavior.

Linking this explanation to the continuous–time HMM decoder takes us back to (55) where the output at a single node is dependent upon a weighted sum of the propagating wave and the current observation match to that node.

The programmable diffuser equation also shows a similar relationship between the current state and previous behaviors. The floating-gate charge, as shown in Figure 76, provides the weighting from previous node to the current node and the magnitude of the input scales the input current at the particular node.

The state transition behavior of HMMs is similar to viewing multiple waves travel along parallel paths with multiple inputs occurring at different places down the path. Depending on the current state of the wave and the input, the wave will either be amplified or attenuated. Outputs taken at the end of these paths is dependent upon the coincidence of a wave phenomena and a series of input sequences.

CHAPTER 6

CONCLUSIONS

Research presented here provides a substantial leap forward in the development of analog computational systems from the study of interesting device physics to fully programmable analog networks and lays a solid foundation for future development of large-scale, compact, low-power analog computation systems. This work builds on earlier investigations of floating-gate devices as computational memory elements. The result of these investigations led to the first fully analog system suitable for auditory feature extraction and recognition.

The core of this research has centered around the floating-gate device as an analog programmable memory element. Extending this concept to computational arrays required methods of accurately programming large numbers of these devices rapidly. Methods for rapid programming, ultra-low programming and various evolutions of programming hardware implementations and algorithms were also presented.

System level investigations started with observation of the analog computational systems for feature extraction and led to the first fully analog programmable band pass filter bank. The programmability was proven through experiments illustrating that each filter bank was programmable and because of the compactness of the filter, making large arrays was fairly straight forward. Furthermore, this investigation identified the key issues associated with implementing higher-order filter topologies including effective input capacitance, Q, input linearity, and programming accuracy for each filter. Results from this system were presented.

With the filterbank as the initial processing block, additional blocks were presented to further process the incoming signal to complete the feature extraction stage. These stages included peak detection, linear voltage-to-current transform and a vector matrix multiplication step. The final feature extraction block was based on a Cepstrum-like processing, but, in the analog domain. Also, core processing blocks required for performing an analog

distance measure and implementing wave propagation networks were introduced. These blocks were the core blocks used in the vector quantization and the hidden-markov model-like decoding blocks respectively.

This research concluded by presenting various signal processing systems that could now be implemented using all of the pieces previously introduced. These systems included a feature extraction block based on cepstrum processing, a vector quantization block and a hidden-markov model-like decoding block. With this system as an initial attempt, it may now be said that analog computational systems can operate cooperatively with digital signal processing systems with this being the example. Analog processing blocks have been used to perform some of the initial processing stages of speech recognition before passing their results to a DSP for further processing. The result being an analog architecture for auditory feature extraction and recognition.

The core concepts in this research are part of the underlying technology in the commercial venture GTronix, Inc.

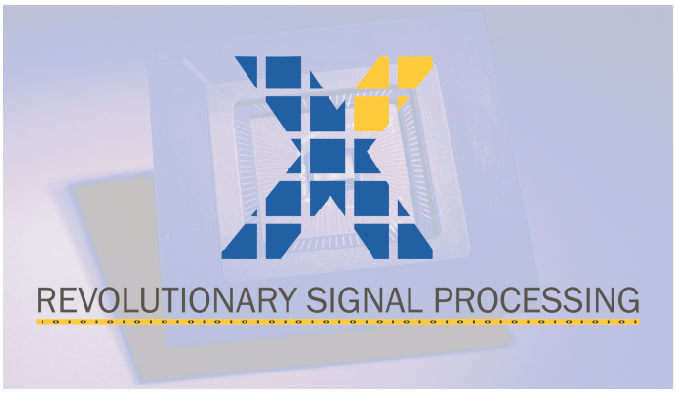
GTroniX, Inc.

Redefining the space of analog circuits through revolutionary signal processing.

Founded, February 2003.

Jeff Dugger, Ph.D.; Paul Hasler, Ph.D.; Matt Kucic, Ph.D.; Paul Smith, Ph.D.

Atlanta, GA



.

REFERENCES

[1] A , F. and H , P., "Offset Removal from Floating-Gate Differential Amplifiers and Mixers.," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, (Tulsa, OK), Aug. 2002.

[2] A , D. V. and H , P., "Cooperative analog/digital signal processing," in *World Conference on Systemics, Cybernetics, and Informatics*, (Orlando, FL), July 2001.

[3] A , A. and B , K., "A 590,000 transistor 48,000 pixel, contrast sensitive, edge enhancing, cmos imager-silicon retina," in *IEEE Symposium on Circuits and system*, 1995.

[4] B , K. and A , A., "A contrast-sensitive retina with reciprocal synapses," in *Advances in Neural Information Processing Systems 4* (M , J., ed.), San Mateo, CA: Morgan Kaufman Publishers, 1991.

[5] C , G. and P , V., "A low-power cmos analog vector quantizer," *IEEE Journal of Solid State Circuits*, vol. 32, pp. 1278–1283, August 1997.

[6] D , T., "Silicon retina with correlation-based velocity-tuned pixels," *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 529–541, 1993.

[7] D ¨ , T. and S , A., "Bias current generators with wide dynamic range," in *Proceedings of the IEEE International Symposium on Circuits and Systems*.

[8] D ¨ , T., "Bump circuits for computing similarity and dissimilarity of analog voltages," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. 1, pp. 475–479, July 8-12 1991.

[9] D , J. R., P , J. G., and H , J. H. L., *Discrete-time Processing of Speech Signals*. New York: Institute of Electrical and Electronics Engineers Press, 2000.

[10] D.K and S , S., "A floating gate and its application to memory devices," *Bell Syst. Tech. J.*, vol. 46, p. 1288, 1967.

[11] D , C., F , E., and H , P., "Practical issues using e-pot circuits," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. V, (Phoenix, AZ), pp. 493 – 496, May 2002.

[12] F , E., "A biologically inspired silicon neuron," Master's thesis, Georgia Institute of Technology, 2003.

[13] F , J. C. and H , K. M., "Dendrite structure," in *Dendrites* (S , G., S - , N., and H , M., eds.), Oxford: Oxford University Press, 1999.

[14] F , G., "Digital signal processor trends," *IEEE Micro*, pp. 52 – 59, Nov 2000.

[15] F , B., W , J., and A , A., "CMOS analog IC implementing the backpropagation algorithm," in *Abstracts of the First Annual INNS Meeting*, vol. 1, p. 381, 1988.

[16] G , A., W , S., and Z , K., "Vector quantization techniques in speech coding," in *Advances in speech signal processing* (F , S. and S , M. M., eds.), pp. 3 – 48, New York: M. Dekker, 1992. Not terribly useful.

[17] G , D. W. and H , P., "Capacitively-coupled current conveyer second-order section for continuous-time bandpass filtering and cochlea modeling," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2002.

[18] H , R. R. and K , C., "An analog VLSI implementation of a visual interneuron: enhanced sensory processing through biophysical modeling," *International Journal of Neural Systems*, vol. 9, pp. 391–395, Oct. 1999.

[19] H , R., B , J., H , P., M , B., , and D , S., "A cmos programmable analog memory-cell array using floating-gate circuit," in *IEEE Transactions on Circuits and Systems Special Issue*, vol. 48, p. 4, January 2001.

[20] H , P., *Foundations of Learning in Analog VLSI*. PhD thesis, California Institute of Techonology, February 1997.

[21] H , P. and L , T., "Overview of floating-gate devices, circuits, and systems," *IEEE Journal of Circuits and Systems II*, pp. 1 –3, January 2001.

[22] H , P., M , B. A., and D , C., "Adaptive circuits using pfet floating-gate devices," in *Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, (Atlanta, GA), pp. 215–229, March 1999.

[23] H , P., M , B. A., and D , C., "An autozeroing floating-gate amplifier," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, pp. 74–82, 2001.

[24] H , P., "Continuous-time feedback in floating-gate mos circuits," *IEEE Journal of Circuits and Systems II*, pp. 56–64, January 2001.

[25] H , P. and A , D. V., "Cooperative analog-digital signal processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. IV, (Orlando, FL), pp. 3972–5, May 2002.

[26] H , P., D , C., M , B. A., and M , C. A., "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7* (T , G., T , D. S., and L , T. K., eds.), pp. 817–824, Cambridge, MA: MIT Press, 1995.

[27] H , P., K , M., and M , B. A., "A transistor-only circuit model of the autozeroing floating-gate amplifier," in *Midwest Conference on Circuits and Systems*, (Las Cruces, NM), 1999.

[28] H , P. and M , B. A., *Floating-Gate Devices, Circuits, and Systems*. IEEE Press, 2002.

[29] H , P., S , P., D , C., G , C., D , J., and A , D., "A floating-gate vector-quantizer," in *IEEE Midwest Circuits and Systems*, (Tulsa, OK), Aug. 2002.

[30] H , P., S , P., E , R., G , D., and A , D. V., "Biologically inspired auditory sensing system interfaces on a chip," in *2002 IEEE Sensors Conference*, vol. 1, (Orlando, FL), pp. 669–674, June 2002.

[31] H , P., S , P. D., F , E., and A , D. V., "A neuromorphic ic connection between cortical dendritic processing and hmm classification," in *DSP Workshop, Taos*, 2004.

[32] H , H. and K , C., "Bifurcations in a vocal fold model,"

[33] H , W. J., "Pitch and voicing determiniation," in *Advances in speech signal processing* (F , S. and S , M. M., eds.), pp. 49 – 84, New York: M. Dekker, 1992.

[34] H , S.-C., I , M., and Z , S. R., "A wide range differential difference amplifier: A basic block for analog signal processing in MOS technology," *IEEE Transactions on Circuits and Systems II*, vol. 40, pp. 289–301, May 1993.

[35] H , M. J., "The speech signal," in *Digital Speech Processing; Speech Coding, Synthesis and Recognition* (I , A. N., ed.), pp. 43 – 71, Boston: Kluwer Academic Publishers, 1992.

[36] K , P., S -S , E., and K , A., "An enhanced adaptive Q-tuning scheme for a 100-mhz fully symmetric ota-based bandpass filter," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 585 – 593, April 2003.

[37] K , J. M., "Accurate tuning curves in a cochlear model," *IEEE Transactions on Speech and Audio Processing*, vol. 1, pp. 453–462, October 1993.

[38] K , J., N , M., and G , P., "A dsp powered solid state audio system," in *icassp*, vol. 4, pp. 2283 – 2286, 1999.

[39] K , M., D , J., H , P., and A , D., "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proceedings of the 21st Conference on Advanced Research in VLSI*, (Atlanta, GA), pp. 148 –162, March 2001.

[40] K , M., L , A., N , J., and H , P., "A programmable continuous-time floating-gate fourier processor," *IEEE Journal of Circuits and Systems II*, January 2001.

[41] K , M., H , P., D , J., and A , D. V., "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *2001 Conference on Advanced Research in VLSI* (B , E. and M , C., eds.), pp. 148–162, IEEE Computer Society, March 2001.

[42] K , M., H , P., D , J., and A , D. V., "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *2001 Conference on Advanced Research in VLSI* (B , E. and M , C., eds.), pp. 148–162, IEEE Computer Society, March 2001.

[43] K , M., L , A., H , P., and N , J., "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 90–99, January 2001.

[44] K , N. and A , G. C. A. G., "Level crossing time interval circuit for micropower analog vlsi auditory processing," in *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, vol. V, pp. 581 – 590, Aug. 1995.

[45] K , N. and A , G. C. A. G., "A circuit model of hair-cell transduction for temporal processing and auditory feature extraction," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 301 – 304, 1996.

[46] L , T. S., *Neuromorphic Systems Engineering, Neural Networks in Silicon*. Kluwer Academic Publishers, 1 ed., 1998.

[47] L , J. and M , C., "A silicon model of auditory localization," *Neural Computation*, vol. 1, pp. 47–57, 1989.

[48] L , J., W , J., and K , A., "Systems technologies for silicon auditory models," *IEEE Micro*, pp. 7 – 15, June 1994.

[49] L , J., W , J., and L , R., "A micropower analog VLSI HMM state decoder for wordspotting," in *Advances in Neural Information Processing Systems 9* (M , M. C., J , M. I., and P , T., eds.), pp. 727–733, Cambridge, Massachusetts: MIT Press, 1996.

[50] L , M. and S , E. H., "Fowler-nordheim tunneling in thermally grown sio_2," *Journal of Applied Physics*, vol. 40, p. 278, 1969.

[51] L -B , V. and S -G , T., "On the design and characterization of femtoampere current-mode circuits," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 1353 – 1363, August 2003.

[52] L , J. and C , G., "A micropower learning vector quantizer for parallel analog-to-digital data compression," in *International Conference on Circuits and Systems*, (Monterey, CA), 1988.

[53] L , R. F., "A computational model of filtering, detection and compression in the cochlea.," in *Proc. of the IEEE Intl. Conf. on Acoust. Speech and Signal Proc.*, (Paris), May 1982.

[54] L , R. and M , C., "An analog electronic cochlea.," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1119 – 1134, July 1988.

[55] M , M. and D , R., "A silicon neuron," *Nature*, vol. 354, no. 6345, pp. 515–518., 1991.

[56] M , M. and M , C., "The silcon retina," *Scientific American*, vol. 264, no. 5, pp. 76–82, 1991.

[57] M , C., *Analog VLSI and Neural Systems*. Massachusetts: Addison-Wesley, 1 ed., 1989.

[58] M , T. K. and S , W. C., *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, NJ 07548: Prentice Hall, 1 ed., 2000.

[59] N , S. T. and K , D. O., "A model for active elements in cochlear biomechanics," *Journal of the Acoustical Society of America*, vol. 79, pp. 1472–1480, 1986.

[60] N , J. D., M , B. K., B , E. A., D W , S. P., and H , P. E., "A cmos coupled nonlinear oscillator array," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. IV, (Phoenix, AZ), pp. 301 – 304, may 2002.

[61] O , A. V. and S , R. W., *Digital Signal Processing*. New Jersey: Prentice-Hall, 1975.

[62] P , Y.-H., *Adaptive Pattern Recognition and Neural Networks*. Oxford University Press, Inc., 1996.

[63] P , G.N.; D W , S., "Analogue vlsi morris-lecar neuron," *Electronics Letters*, vol. 33, pp. 997 – 998, June 5 1997.

[64] P , T. K. and A , P. E., "A highly accurate step-response-based successive-approximation frequency tuning scheme for high-q continuous-time bandpass filters," *IEEE Transactions on Circuits and Systems I*, vol. 50, pp. 221 – 227, May 2003.

[65] R , L., "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[66] R , S., M , N., B , H., C , M., and F , H., "Connectionist probability estimators in hmm speech recognition," in *IEEE Transactions On Speech And Audio Processing*, vol. 2, pp. 161–174, January 1994.

[67] S , C. and S , R., "A practical micropower programmable bandpass filter for use in bionic ears," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 63 – 70, January 2003.

[68] S -S , E. and S -M , J., "CMOS transconductance amplifiers, architectures and active filters: a tutorial," in *IEE Proceedings - Circuits, Devices and Systems*, vol. 147, pp. 3 – 12, February 2000.

[69] S , R., *Efficient precise computation with noisy components: extrapolating from an electronic cochlea to the brain*. PhD thesis, California Institute of Technology, Pasadena, CA, 1997.

[70] S , R., "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, pp. 1601–1608, 1998.

[71] S , A. V., F , E., and V , E., "Improved silicon cochlea using compatible lateral bipolar transistors," in *Advances in Neural Information Processing Systems 8* (T , D., ed.), (Cambridge, MA), pp. 671–677, MIT Press, 1996.

[72] S , J., *Pattern Classification, A Unified View of Statistical and Neural Approaches*. New York: John Wiley and Sons, Inc., 1996.

[73] S , G., S , P., L , H., C , R., H , T., T , C., and P.H , "Automatic rapid programming of large arrays of floating-gate elements," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Vancouver, Canada), 2004. Accepted.

[74] S , K. and I , F., "Spectrum distance measures for speech recognition," in *Advances in speech signal processing* (F , S. and S , M. M., eds.), pp. 419 – 452, New York: M. Dekker, 1992.

[75] S , P. and H , P., "The analog speech recognition project," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, (Orlando, FL), pp. 3988–3991, May 2002.

[76] S , P. and H , P., "A programmable diffuser circuit based on floating-gate devices," in *Midwest Circuits and Systems*, (Tulsa, OK), Aug. 2002.

[77] S , P., K , M., E , R., H , P., and A , D. V., "Mel-frequency cepstrum encoding in analog floating-gate circuitry," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. IV, (Phoenix, AZ), pp. 671–674, May 2002.

[78] S , P. D., G , D. W., C , R., and H , P., "A five-transistor bandpass filter element," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Vancouver, Canada), 2004. Submitted.

[79] S , P. D., H , P., and A , D. V., "Analog speech recognition project," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, (Orlando, FL), pp. 3988–3991, 2002.

[80] S , P. D., K , M., and H , P., "Accurate programming of analog floating-gate arrays," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, (Phoenix, AZ), pp. 489–492, May 2002.

[81] T , C., "Calculation of TMS320LC54x power dissipation," Application Report SPRA164, Texas Instruments, Digital Signal Processing Solutions - Semiconductor Group, June 1997.

[82] W , M., H , P., and A , L., *CMOS Neural Networks for Pattern Association*. Special Feature, IEEE Micro, October 1989.

[83] W , K. and S , S. A., "A functional model of the early auditory system.," in *Proceedings of the IEEE-SP International Symposium, Time-Frequency and Time-Scale Analysis*, pp. 45 – 48, Oct 4-6 1992.

[84] W , L., K , D., L , R., and M , C., "Improved implementation of the silicon cochlea.," *IEEE Journal of Solid-State Circuits*, pp. 692 –700, May 1992.

[85] W , M., O , M., M , M., and S , E., "Speech recognition in analog multichannel cochlear prostheses: initial experiments in controlling classifications.," in *IEEE Transactions on Biomedical Engineering*, pp. 1002 –1010, Oct 1990.

[86] W , E. and S , S., "Comparison of linear prediction cepstrum coefficients and mel-frequency depstrum coefficients for language identification," in *International Symposium on Intelligent Multimedia, Video and Speech Processing*, (Hong Kong), pp. 95 – 98, May 2001.

[87] Z , M., "Comparison of four approaches to automatic language identification of telephone speech.," in *IEEE Transactions on Speech and Audio Processing*, vol. 4, pp. 31–44, January 1996.